ESD-TR-82-056

Technical Report

612

V.J. Sferrino

A Multiport Buffer Memory for an Internet Packet Voice/Data Gateway

14 July 1982

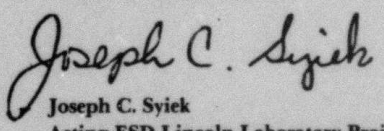
Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-80-C-0002 by

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-80-C-0002 (ARPA Order 3673).



MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

A MULTIPORT BUFFER MEMORY FOR AN INTERNET PACKET VOICE/DATA GATEWAY

V.J. SFERRINO
Group 24

TECHNICAL REPORT 612

14 JULY 1982

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes the design of a Multiport Buffer Memory to be used in an internet packet voice/data gateway for an experimental wideband satellite network. The gateway consists of a central processor and a number of microprocessor-controlled interfaces to local networks at each site and to the satellite network. The Multiport Buffer Memory is accessible from any of these microprocessors, and increases system throughput by allowing the bulk of the packet traffic between the networks to pass through it without having to pass through the central processor.

The shared memory consists of a main memory organized as 64K words x 16 bits and an auxiliary pointer memory of 4K words x 16 bits. The Multiport Buffer Memory complex includes an automated intelligent memory management strategy which makes the memory resource allocation transparent to the microprocessor-based systems which share the memory.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

A MULTIPORT BUFFER MEMORY

FOR AN INTERNET PACKET VOICE/DATA GATEWAY

1. INTRODUCTION

An experimental wideband satellite-based packet system[1] is being implemented to develop and demonstrate techniques for achieving the advantages of integrating packet voice and data in a realistic large scale network. The experiment is designed around a satellite channel with a capacity of 3 Mbps which will be capable of supporting many simultaneous voice connections. The experimental system consists of research facilities at multiple sites which are linked by a wideband packet satellite network (the WB SATNET) including a channel on a satellite transponder, earth stations, high-performance burst modems, and demand assignment multiple access (DAMA) processors. Access from various packet speech facilities onto the WB SATNET is obtained by means of an internetwork packet voice/data gateway at each site. The gateway is composed of a central processor and a number of microprocessors controlling the interfaces to local networks and the WB SATNET. A primary function of the gateway is to multiplex the packet voice/data traffic onto the satellite channel. In the current gateway configuration, all packet traffic must pass through the central gateway processor. This represents a throughput limitation on the overall system. A special subsystem, the Multiport Buffer Memory (MBM), has been designed to allow the digital voice traffic to pass from local nets to the WB SATNET without passing through the central gateway processor. The MBM allows the speech portions of digitized voice packets to be transferred, under control of the central gateway processor, directly between the various microprocessors servicing each net. This report describes the design of the MBM.

---

[1]C.J.Weinstein and H.M.Heggestad, " Multiplexing of Packet Speech on an Experimental Wideband Satellite Network", AIAA 9th Communications Satellite Systems Conf., San Diego, California, 7-11 May 1982, pp. 115-124.

This introductory section includes a presentation of the rationale for the overall design concept as well as some elements of the system functional operation. Section 2 presents the detailed design and functional description of the major portions of the bussed architecture. Section 3 describes the Memory Management Logic which includes the organization of the Main and Auxiliary Memory complexes. Sections 4 and 5 detail the memory read and write operations. Section 6 describes the Finite State Machine which effects the control of the bussed architecture. Section 7 describes the Bus Arbitration Logic which grants accesses to the various ports attached to the MBM. Section 8 deals with various aspects of the engineering design of the MBM.

A block diagram of the system is shown in Fig. 1. The central gateway processor for this system is the PDP-11/44 which serves as a *Miniconcentrator* host for a cluster of subsystems attached to the Miniconcentrator via UMC Z80 microprocessor boards. These boards were designed by Associated Computer Consultants to provide independent processing capability on the PDP/11 UNIBUS. Various subsystems can be attached through I/O ports on the UMC Z80s and Fig. 1 shows a typical configuration of a subsystem complex. This represents the subsystem complex at the Lincoln Laboratory site of the WB SATNET. Similar multiprocessor configurations exist at other sites around the country including ISI, SRI, and DCEC. There are various other possible configurations at these sites which include such subsystems as the Packet Radio Net at SRI and the Experimental Data Network at DCEC. The subsystems at the Lincoln Laboratory site include the LEXNET terminal complex, the Pluribus Satellite IMP (PSAT), the Packet-Circuit Interface (PCI), and ARPANET gateway. Data from each subsystem are regulated by the host for single stream packetized communication over the satellite channel. The data rates for each channel are such that throughput is a critical issue and packet storage management limits the overall throughput when packets must be passed through the system and stored in PDP-11 Memory.

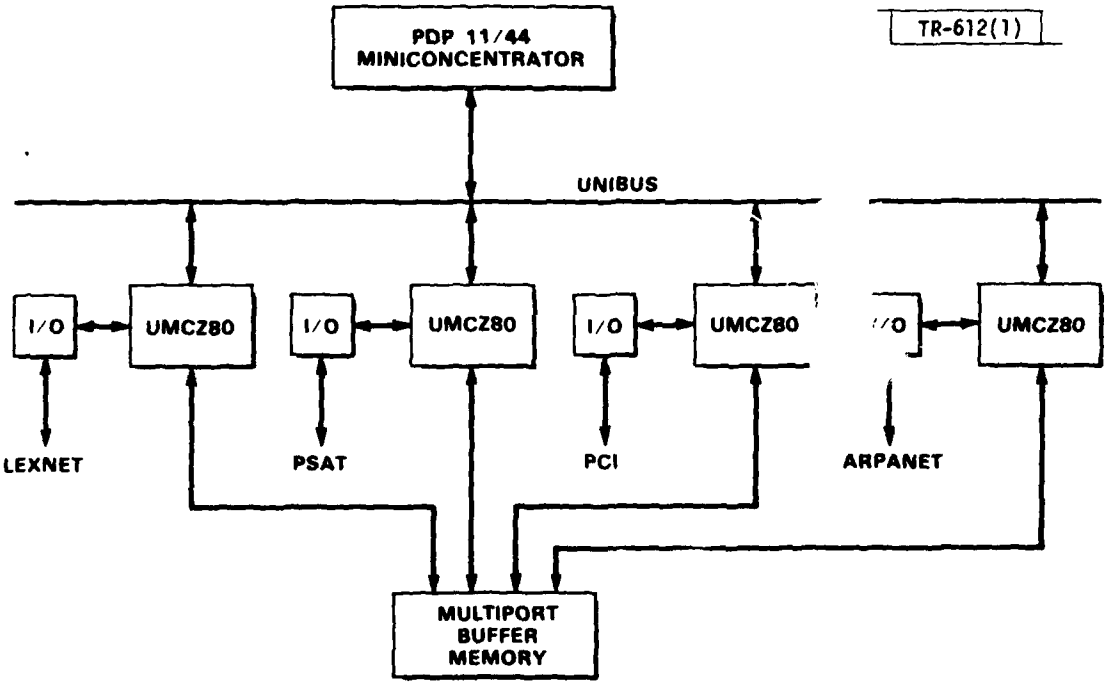


Fig. 1. System block diagram.

The storage management overhead imposed upon the Miniconcentrator for such packet transfers is considerable, but the addition of the MBM improves the total throughput capability of the WB SATNET by providing a shared memory resource which is attached to all of the UMC Z80s. The memory is directed by an intelligent controller which manages its own resource allocation and makes that function transparent to the entire community of users thereby relieving the Miniconcentrator of the overhead for block manipulation which would otherwise be necessary without the shared memory. The quantification of throughput improvement is dependent upon many factors and is constantly changing as the system design evolves. The greatest throughput improvement is realized in an unbalanced traffic environment which requires a large degree of block manipulation for which the MBM provides a common pool of buffer management for block transfers. The savings which are effected through the use of the MBM stem from the intelligent memory management which resides in the architecture of the shared memory.

The Memory Management Logic employs a strategy which traces the location of stored strings of data blocks in Main Memory by means of pointers to those blocks which are maintained in a separate Auxiliary Pointer Memory. Data to be stored or retrieved in Main Memory are accessed by starting location and word length parameters only, thereby relieving the Miniconcentrator of ongoing block manipulations for long packets of data. These parameters are circulated as part of the header information for the entire packet in the Miniconcentrator environment and are used in communicating between the UMC Z80s and the shared memory.

The scenario for the systems environment of the MBM involves the flow of packets of data bytes between each of the functional subgroups that comprise the Miniconcentrator complex. A typical packet for speech communication using a PCM algorithm would consist of approximately 160 bytes. For such a data

packet a typical receive transaction currently takes 2.39 ms without the MBM. It is estimated that the use of the MBM would reduce the receive operation to 1.33 ms. These numbers assume no waiting time in the UMC Z80 during UNIBUS transactions with the Miniconcentrator and would have to be adjusted to reflect that. Although no quantitative estimate has been made of the impact of this factor, PDP-11/44 memory is currently used for moving packets through the network and this would add even more to the current packet transit time and thereby result in a further enhancement of the throughput when using the MBM. In a balanced traffic environment not much block manipulation is encountered. However for unbalanced one-way traffic the buffer length begins to grow dangerously, and it is in this special environment that the MBM offers the greatest overall throughput advantage. The system efficiency is improved through the addition of the MBM by providing a common pool of buffer storage locations instead of a number of individual buffers which would waste buffer space in some applications and would run out of buffer space in other situations.

The basic architectural configuration of the MBM is shown in Fig. 2. Each UMC Z80 bus is attached to a set of Input and Output Latches which convert the 8-bit bytes of the UMC Z80 to 16-bit words in the MBM. The byte conversion is consistent with the normal packet traffic in the network and further serves to supply the additional time required to perform the functional operations which include bus arbitration between the 4 ports and memory management. The use of 16-bit words in the MBM design was also chosen in anticipation of the possible long term conversion to 16-bit microprocessors for the Network Interface Processors. This would allow the MBM to be used with such processors by simply eliminating the 8 to 16-bit conversion logic. The basic architecture of the MBM is comprised of a data bus and an address bus to which is attached the Main and Auxiliary Memory complexes, as well as a set of tri-state registers which accomplish the memory management and bus arbitration functions.

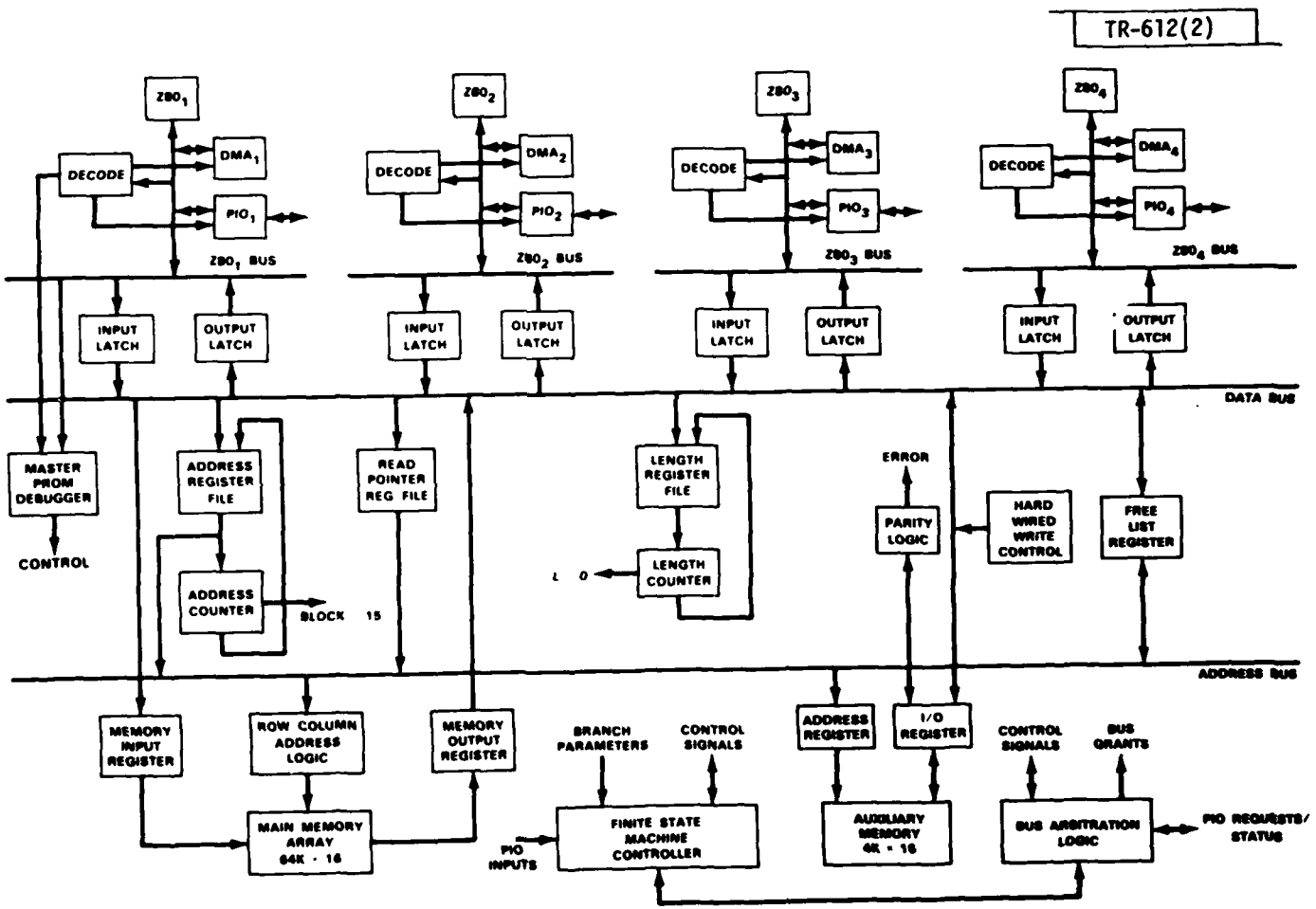


Fig. 2. Basic architectural configuration.

The total memory capacity available to be shared among the 4 UMC Z80 ports attached to the MBM is 64K x 16. A Z80 DMA is attached to each external UMC Z80 bus for controlling fast block transfers. A typical DMA cycle in the Z80 requires 5 clock cycles or 1.25 us. Since the data word transfers are effected in 8-bit bytes then two byte transfers are necessary for each 16-bit word in the Main Memory. This allows at least 2.5 us during which access may be afforded to to all 4 ports requesting it, and thereby allows sufficient time for each one to perform a Main Memory read or write before any slowdown occurs in DMA transfers due to the shared memory. If a completely transparent design is hypothesized wherein simultaneous access is desired in the worst case for all four complexes, then a totally interleaved arbitration scheme may be achieved if each UMC Z80 is allotted 625 ns in which to access memory. A dedicated Parallel Input/Output Controller (PIO) is provided for each UMC Z80 whose function is to supply status information to the UMC Z80 and to issue control bytes from the UMC Z80 for setting up memory read and write operations as well as engendering interrupts for special conditions. The PIO is attached directly to the external bus of each UMC Z80 . Each external bus also includes logic on the address lines to decode a particular set of reserved addresses which signal communication between the MBM and the UMC Z80. The data bus is composed of a complex of tri-stated registers which effect the Memory Management Logic. A Finite State Machine (Fig. 3) controls the Memory Management Logic. This is implemented as a set of PROMs which are addressed by a combination of branching parameters derived from the logic as well as next state pointers which are programmed into the PROMs to effect the flow of operations necessary to perform the read and write operations in the Main Memory. This design was selected in place of a microprocessor controller because investigation of the available microprocessors (either bit-slice or single-chip) revealed that none of these devices was fast enough to permit full control of the the architecture within the 625 ns memory

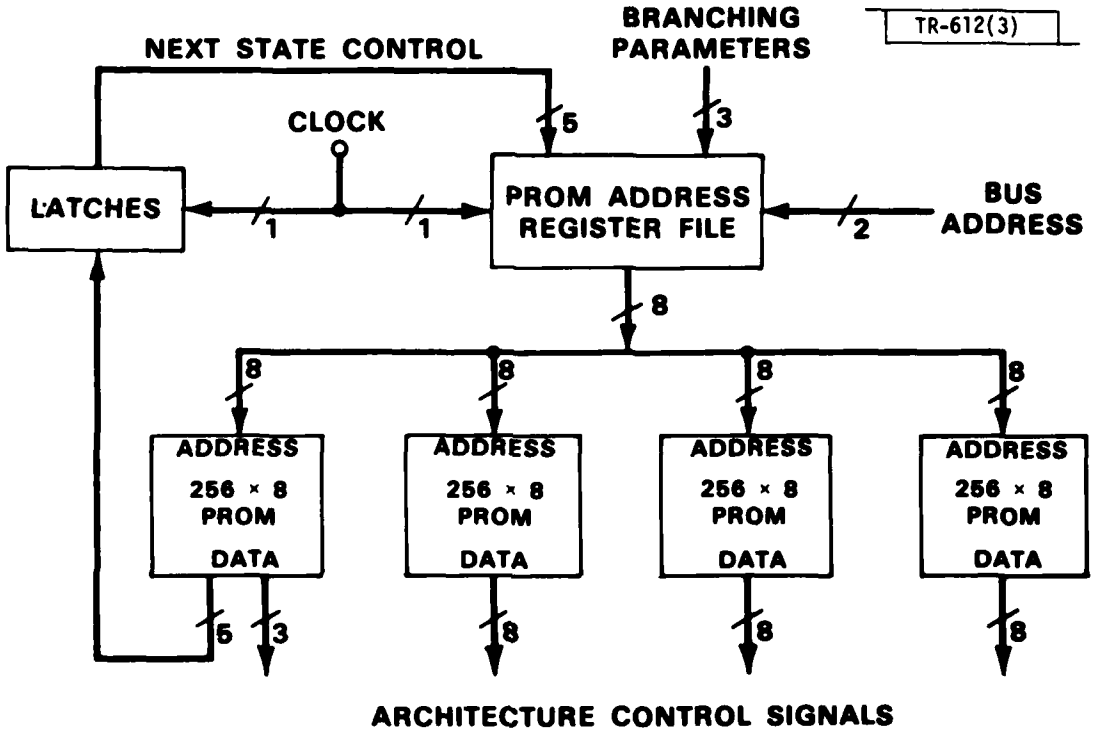


Fig. 3. Finite state machine.

cycle allotted for each port access. The Finite State Machine is, in effect, a special-purpose microprocessor which is programmable by virtue of changes in the content of the PROMs. This allows functional changes to be made in the data flow within the architectural design as the system testing evolves and is well suited to the environment of the WB SATNET. The control of the MBM is overlaid by the Bus Arbitration Logic. This logic is also implemented in PROM in order to allow the most rapid operation of the system so that it does not slow down any UMC Z80 complex beyond the normal DMA rate in the absence of the shared resource.

The Main Memory complex consists of a bank of 16-64K x 1 dynamic RAMs (MCM6664) which provides 64K x 16 bits of memory. An Auxiliary Pointer Memory comprised of 4-4K x 4 static RAMs (IMS1420) is used to implement the Main Memory storage management. The management of the 64K memory address space is governed by the necessity to intersperse data packet storage locations as bus grants are assigned to the various requesting ports by the Bus Arbitration Logic, and to keep track of their location so that each port can access that data by using only a single address pointer to represent the entire packet. In writing a block of memory data, any UMC Z80 provides a packet word length to the MBM which then returns the address of the first location in Main Memory in which that packet is to be stored. The UMC Z80 then attaches the address pointer to the header of the packet before passing it on to the PDP-11/44 for system traffic control. The packet then carries with it the location of the first pointer address to the data string without having to retain any knowledge of how that string is distributed in the Main Memory. A read operation is effected by providing a packet word length to the interface as well as a pointer which represents the address of the starting location of the desired packet in Main Memory. The Auxiliary Pointer Memory is used as a mechanism to store the address chain of all 16 word blocks which comprise the entire packet of data in the Main Memory. Since

the block size of each string in Main Memory is 16 words and since the total number of words is 64K, then 4K words of memory address are required to address those blocks. In principle the design can accommodate any block size which is a power of 2, but the 16-word block size was chosen as a reasonable compromise for the expected block manipulation scenario. Other block sizes would affect the size of the Auxiliary Pointer Memory and the block pointer word length. Although only 12 bits are required to provide an address for the 4K blocks of 16 words, the Pointer Memory contains a total of 16 bits. The remaining 4 bits are used to provide end-of-string and buffer-full flags as well as providing a parity check on the Auxiliary Memory data, since any errors in the Pointer Memory can have potentially disruptive effects on the system unless ongoing operations are aborted upon detection of parity errors during read operations in the Auxiliary Pointer memory. Fig. 4 shows the conceptual Main Memory organization which consists of 4K blocks each containing 16 words. Fig.5 shows one possible configuration for an initialization of the Auxiliary Pointer Memory. The initialization of this memory is directed from one of the UMC Z80s attached to the MBM, and any set of initial conditions must consist of a chained list of address pointers to 16-word blocks in the Main Memory. Since data blocks from any port attached to the shared memory complex may be totally interleaved, the initialized chain of pointers will be revised to reflect the rearrangement of strings due to ongoing read and write operations. The varying creation and deletion of packet strings due to write and read operations in the memory results in a *free list* of pointers to blocks that have been released after a read operation that does not require the restoration of read data. Read operations may be of two varieties: read/restore or read/release. For a read/restore operation the current free list location is unaltered after the operation, whereas in the case of a read-release operation the Memory Management Logic alters the free list to accommodate the added locations available for writing data due to the released

TR-612(4)

12-BIT BLOCK ADDRESS	4-BIT WORD ADDRESS	16-BIT DATA WORD
0	0 ↓ 15	DATA BLOCK 1 (16 Words)
1	0 ↓ 15	DATA BLOCK 2 (16 Words)
2	0 ↓ 15	DATA BLOCK 3 (16 Words)
3	0 ↓ 15	DATA BLOCK 4 (16 Words)
≈		≈
4095	0 ↓ 15	DATA BLOCK 4096 (16 Words)

Fig. 4. Conceptual main memory organization.

POINTER MEMORY ADDRESS	BITS 4-15	BIT 3	2	1	0
0	1	0	0	0	1
1	2	0	0	0	1
2	3	0	0	0	0
3	4	0	0	0	1
4	5	0	0	0	0
5	6	0	0	0	0
6	7	0	0	0	1
7	8	0	0	0	1
8	9	0	0	0	0
9	10	0	0	0	0
10	11	0	0	0	1
11	12	0	0	0	0
	≈				≈
4094	4095	0	0	0	0
4095	0	0	1	0	1

BIT 0 = PARITY BIT (Even)
BIT 1 = STRING TERMINATOR
BIT 2 = BUFFER-FULL FLAG
BIT 3 = SPARE

117673-N

Fig. 5. Initialized pointer memory contents.

string. The Free List Register is initialized to point to the first block in Main Memory when nothing has been previously stored. The Free List Register is used as a single location register to store the address of the current free block to be written in Main Memory.

2. DETAILED ARCHITECTURE

The architecture shown in block diagram form in Fig. 2 consists of a number of distinct functional elements which comprise the MBM design. This section discusses the detailed design of each element and its functional operation.

2.1. Parallel Input/Output Controller (PIO)

Each UMC Z80 external bus has a Z80 PIO attached to it. The PIO is a programmable dual-port device that provides a means of conveying information regarding the status of the MBM back to the UMC Z80 which is programmed to act upon that status information, as well as a mechanism for issuing control signals. The two ports of the PIO are assigned such that Port A contains a number of status bits which can (under UMC Z80 program control) be strobed to sense the status parameters. Selected status bits can be programmed to engender an interrupt to the UMC Z80 upon the detection of special status conditions. The second port (Port B) consists of output bits which provide either a set of control functions which convey programmed requests for normal read or write operations, or a special set of controls which enable a particular UMC Z80 (subject to bus arbitration) to step data through the various architectural subdivisions of the MBM. Fig. 6 presents a listing of the functional assignments of the input status bits on Port A and the output control bits on Port B. The ability of the PIO to interrupt the CPU of the UMC Z80 on the state of any of the port bits reduces the time the processor must otherwise spend in polling peripheral status.

(I) CHANNEL A - INTERFACE TO Z80

BIT	
0	WRITE POINTER READY
1	END OF READ
2	END OF WRITE
3	PARITY ERROR
4	STRING ERROR
5	BUFFER FULL
6	SPARE
7	SPARE

(II) CHANNEL B - Z80 TO INTERFACE

BIT	
0	READ/RESTORE REQUEST
1	READ/RELEASE REQUEST
2	READ POINTER READY
3	LENGTH READY
4	DEBUG ENABLE
5	WRITE REQUEST
6	RESET
7	SPARE

Fig. 6. PIO functional bit assignments.

2.2. Direct Memory Access (DMA)

Each UMC Z80 external bus has a Z80 DMA attached to it. These dedicated DMAs are used to provide either a read or a write operation in the fastest possible mode between the MBM and each of the attached UMC Z80s. The DMA provides a means of conducting CPU-independent transfers between two ports which, in this case, are the UMC Z80 bus and the MBM . The DMA is armed by the UMC Z80 for either read or write operations in the MBM. The setup of these data-oriented transfers is preceded by all necessary transactions between the MBM and the UMC Z80s as governed by status and control bits in the associated PIOs. Handshaking is provided by the logic shown in block diagram form in Fig. 7 which generates a DMA ready when all setup conditions have been satisfied. Acknowledgement of a DMA transfer is provided by the DMA and the Bus Arbitration Logic controls the allocation of new DMA transfers for each UMC Z80. The Finite State Machine controls the DMA transfer times for the read and write operations.

2.3. Address Decoding

Each UMC Z80 external address bus has attached to it a set of special Decoder Logic (shown in Fig. 8) which decodes the address bits on the bus from a set of reserved addresses which serve to control the tri-stating of those registers which are to be involved in the transfer of data across the data bus. Fig. 9 shows the address assignments which are uniquely reserved for communication to and from the MBM, and the functional meaning of those addresses. All UMC Z80s use the same decoded addresses so that common software can be used in the UMC Z80s . Bus Arbitration Logic is responsible for steering the selected devices to the shared data and address busses.

2.4. Master PROM Debugger

The issue of debugging the MBM complex is a critical one which is addressed through the use of a special PROM-based Debugger. The debugger is

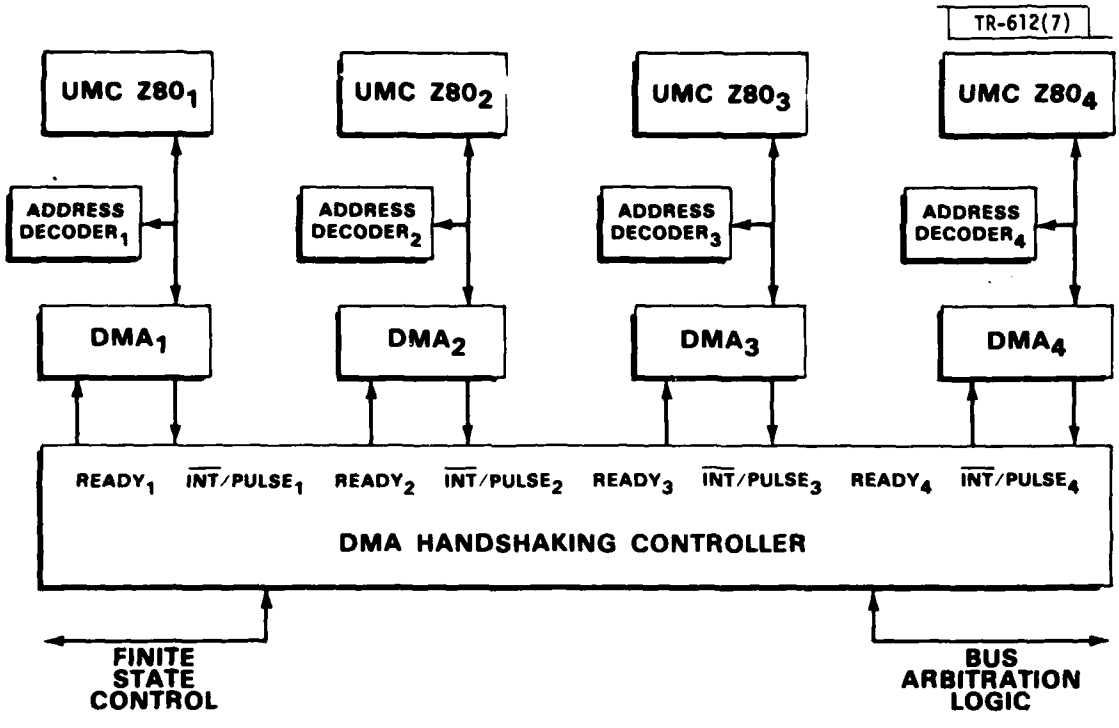


Fig. 7. DMA handshaking logic.

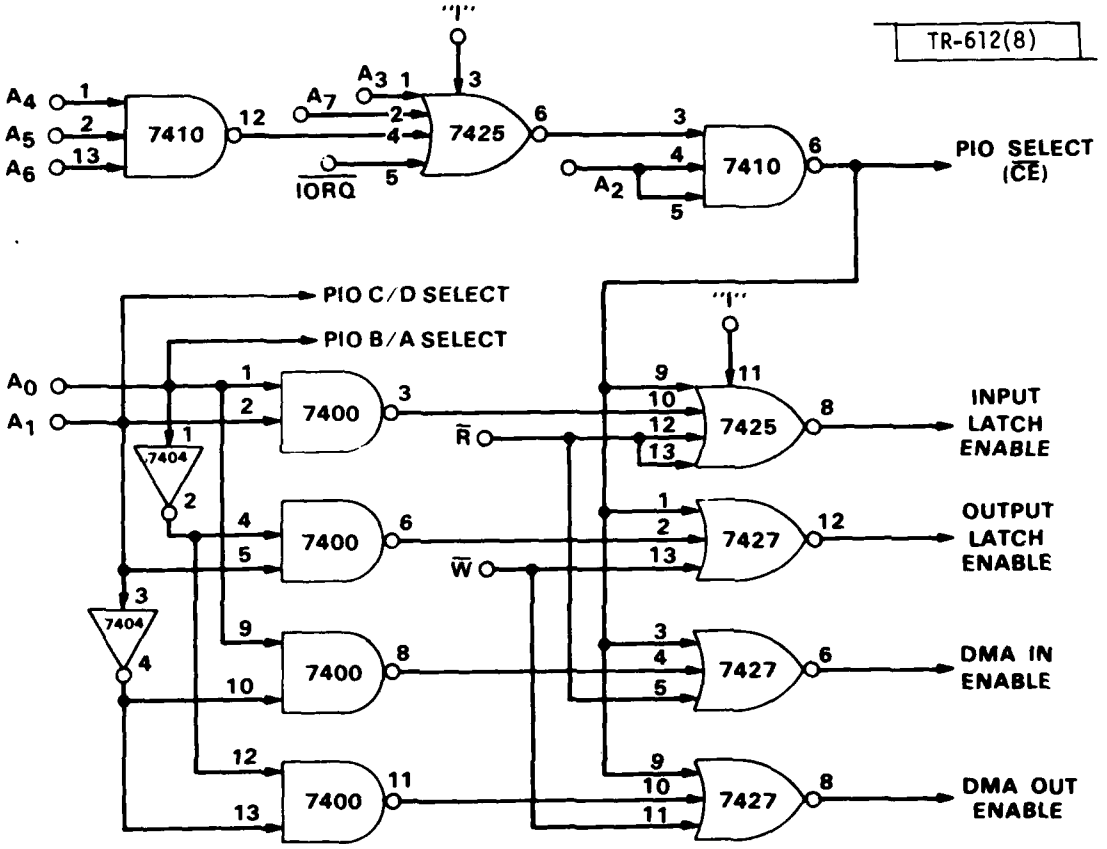


Fig. 8. Address bus decode logic.

TR-612(9)

ADDRESS (Decimal)	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	FUNCTION
119	0	1	1	1	0	1	1	1	PIO CONTROL B
118	0	1	1	1	0	1	1	0	PIO CONTROL A
117	0	1	1	1	0	1	0	1	PIO DATA B
116	0	1	1	1	0	1	0	0	PIO DATA A
115	0	1	1	1	0	0	1	1	INPUT LATCH
114	0	1	1	1	0	0	1	0	OUTPUT LATCH
113	0	1	1	1	0	0	0	1	DMA IN
112	0	1	1	1	0	0	0	0	DMA OUT
111	0	1	1	0	1	1	1	1	MASTER PROM DEBUGGER

Fig. 9. Address bus assignments.

implemented using a set of four 256 x 8 PROMs which are attached directly to the data bus of a UMC Z80 to provide an 8-bit address for the PROMs. The PROMs provide a set of control signals which can direct the flow of data within the MBM architecture. The PROMs are selected by a decoded address on the UMC Z80 address bus. Only one of the four UMC Z80s attached to the data bus is provided with this feature and, therefore, it is deemed the *Master PROM Debugger*. The outputs of the PROM are multiplexed with all normal control functions for all registers of the MBM architecture, and are given priority over the normal bus arbitration function such that the Master Debugger UMC Z80 can force specified data transfers in the architecture to occur. The normal function of bus arbitration is disabled when the Master Debugger is in operation. The Master PROM Debugger facility is used to provide the initialization of the Auxiliary Pointer Memory. Fig. 10 shows a block diagram of the Master PROM Debugger.

2.5 Input/Output Latch Complex

Each UMC Z80 external data bus is connected to a set of Latches (shown in Fig. 11) which serve to buffer the data transfers between the UMC Z80s and the MBM. The Latches are implemented using tri-state octal transparent latch devices (74LS373). Control of the latch data transfers between the UMC Z80 and the MBM is effected by signals which are decoded directly from the external UMC Z80 address bus. Single-byte transfers are effected by input and output instructions when passing setup parameters for memory read and write operations and by DMA control when passing data. All UMC Z80 bytes are 8 bits wide and thus it is necessary to pass two UMC Z80 bytes to or from the interface for each read or write operation in the Main Memory complex in order to be compatible with the 16-bit word architecture of the MBM. There are two Latches for input data and two Latches for output data which provide this conversion. Control of the switching of bytes between these two Input and Output Latch pairs is effected by Odd/Even Control Logic as shown in Fig. 12. A reset control bit on the PIO is used

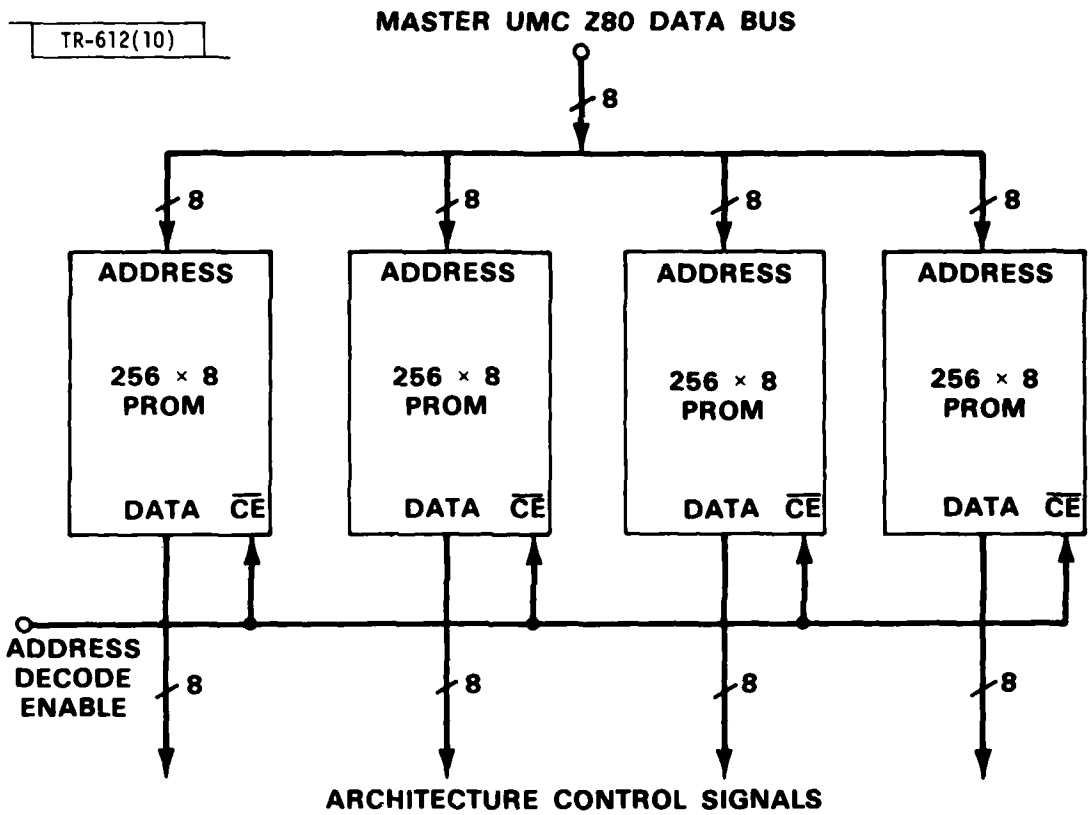


Fig. 10. Master prom debugger.

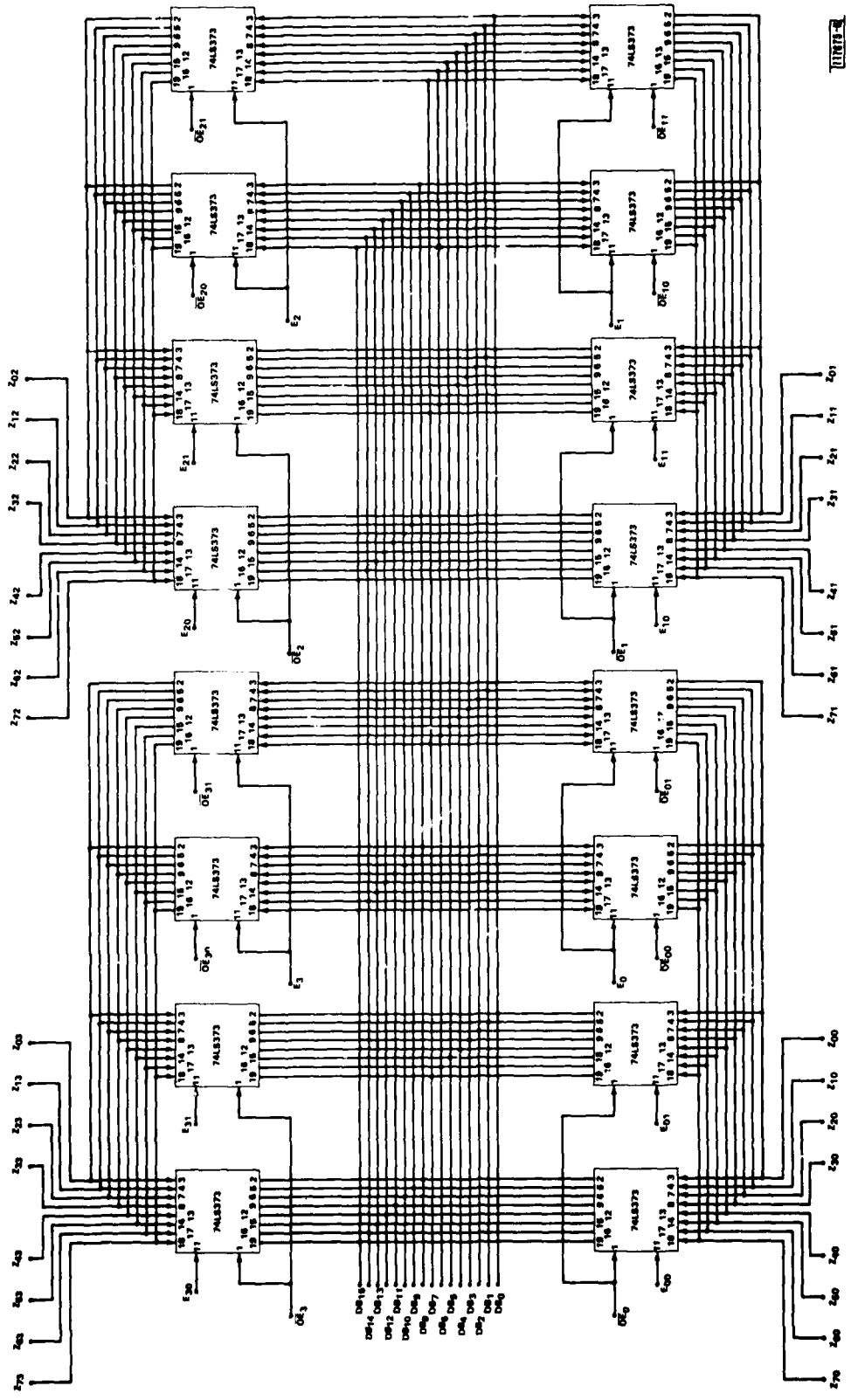


Fig. 11. Input/output latch complex.

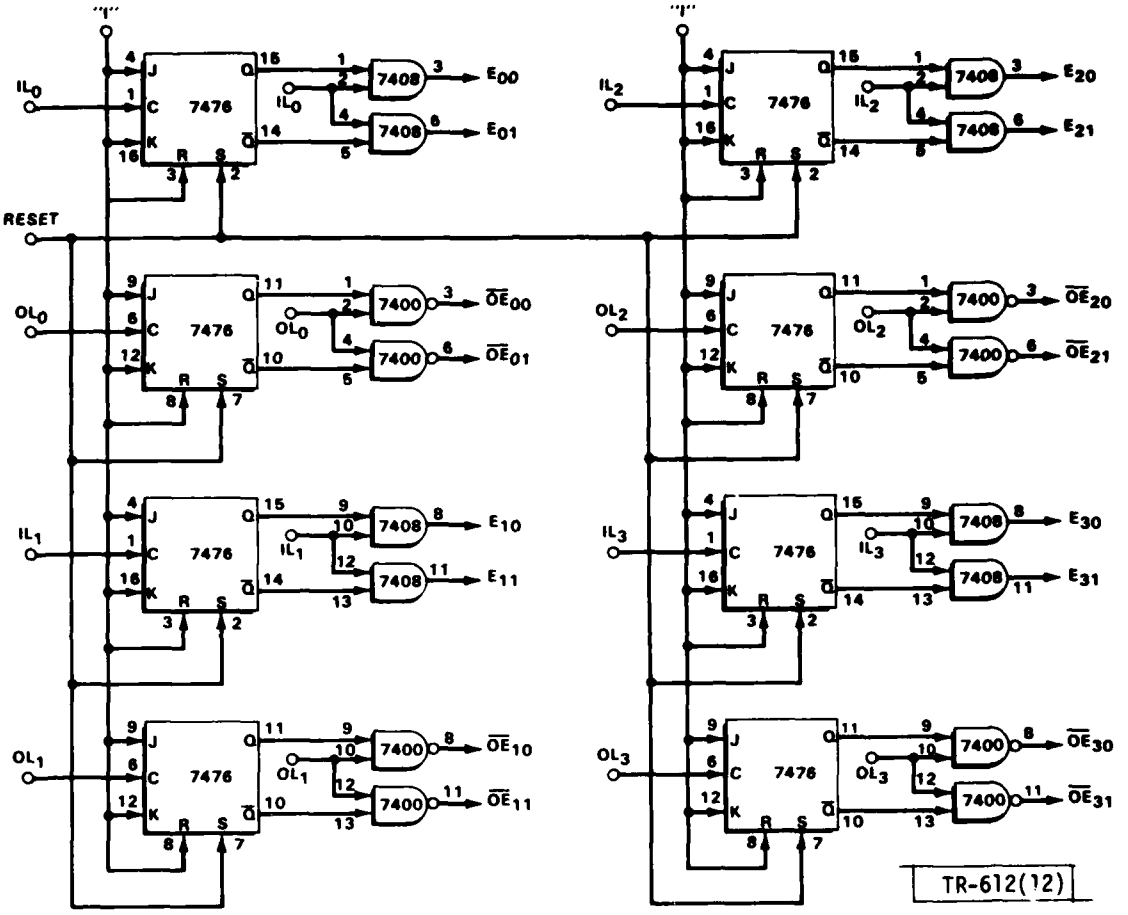


Fig. 12. Odd/even control logic.

to initialize this logic as well as all other register startup conditions.

2.6. Main Memory Complex

The Main Memory array of the MBM is shown in Fig. 13. The memory array is comprised of a set of 16 dynamic RAM elements (MCM6664) which are organized as 64K x 1 bit. This provides a Main Memory array of 64K words by 16 bits. The choice of dynamic RAM was driven primarily by the limited space allotted to the full MBM design. These devices are 16-pin DIPs which utilize row/column addressing to conserve pins. A single 74LS606 octal 2-input multiplexed latch device provides a convenient control element for the application of the row/column address sequence required by the dynamic RAMs by multiplexing the upper and lower 8 bits of the address onto the 8 address inputs of those devices. Figs. 14 and 15 show the Main Memory Input and Output registers which are used to tri-state the memory inputs and outputs to the data bus.

The read operation timing for the MCM6664 dynamic RAMs is shown in Fig. 16. The specification for the read/write cycle time of these devices is 350 ns and, therefore, the 625 ns allotted for the full memory cycle assures a reliable environment for memory timing. The remainder of the cycle time is utilized to provide a memory refresh as part of every memory access, and to make use of the internal capability in the MCM6664 for automatic refresh such that no additional throughput penalty is assessed for this special function of the dynamic RAM chips. The write timing for the Main Memory complex is shown in Fig. 17.

2.7. Auxiliary Pointer Memory Complex

The Auxiliary Pointer Memory array is shown in Fig. 18. The memory is comprised of a set of 4-4K x 4 static RAM chips (IMS1420). These static RAMs are used to store the address pointers to the 4K blocks of 16 words that constitute the Main Memory complex. The 12 most-significant bits of the Auxiliary Pointer Memory are used to address the upper bits of the main address complex, while the 4 least-significant bits are composed of a parity bit, end-of-string bit,

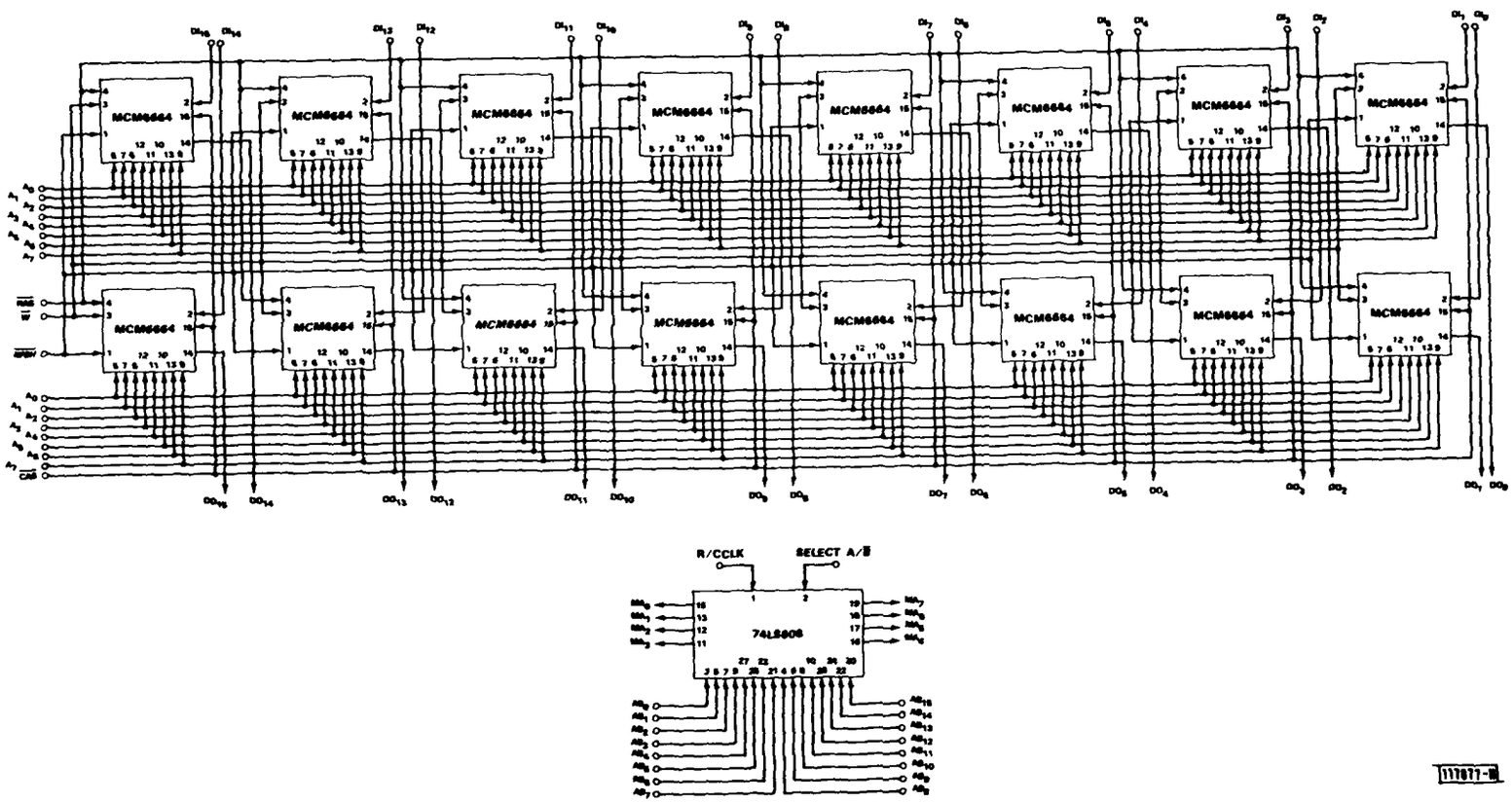


Fig. 13. Main memory complex.

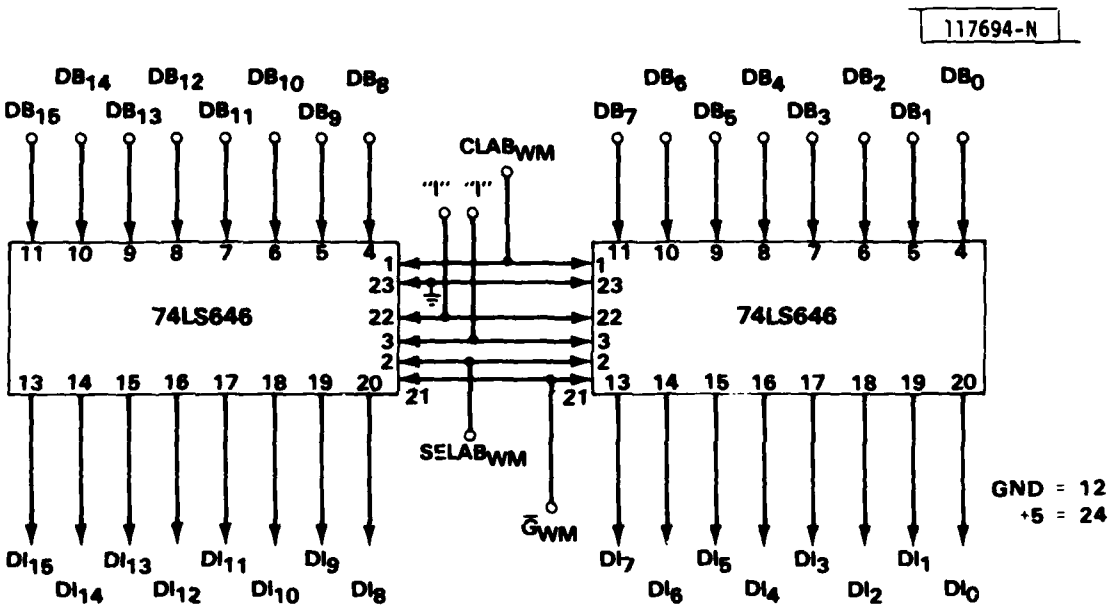


Fig. 14. Main memory input register.

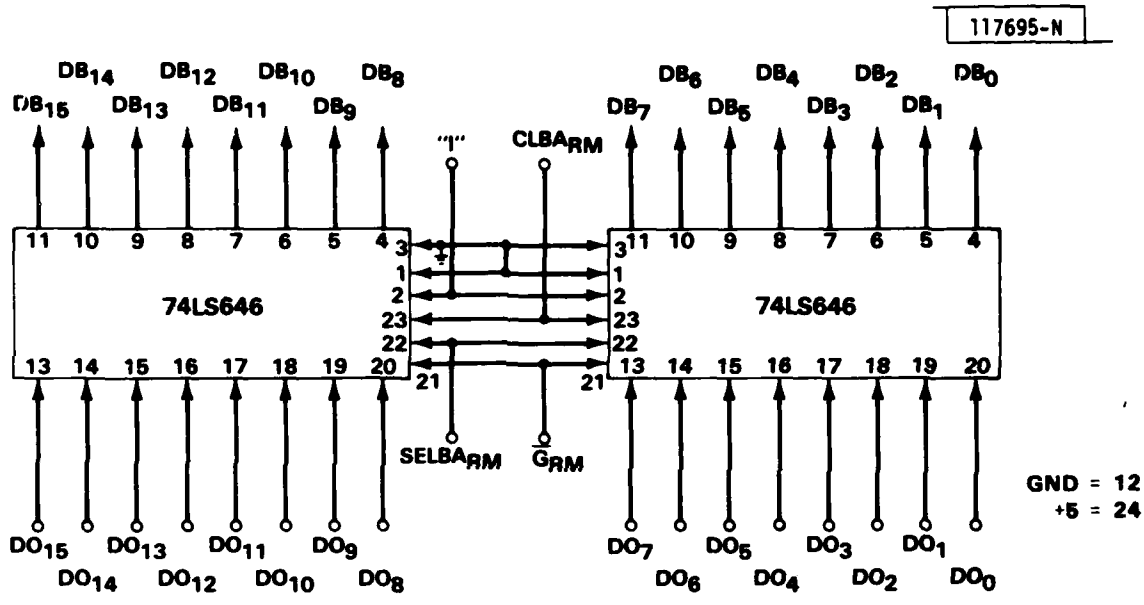


Fig. 15. Main memory output register.

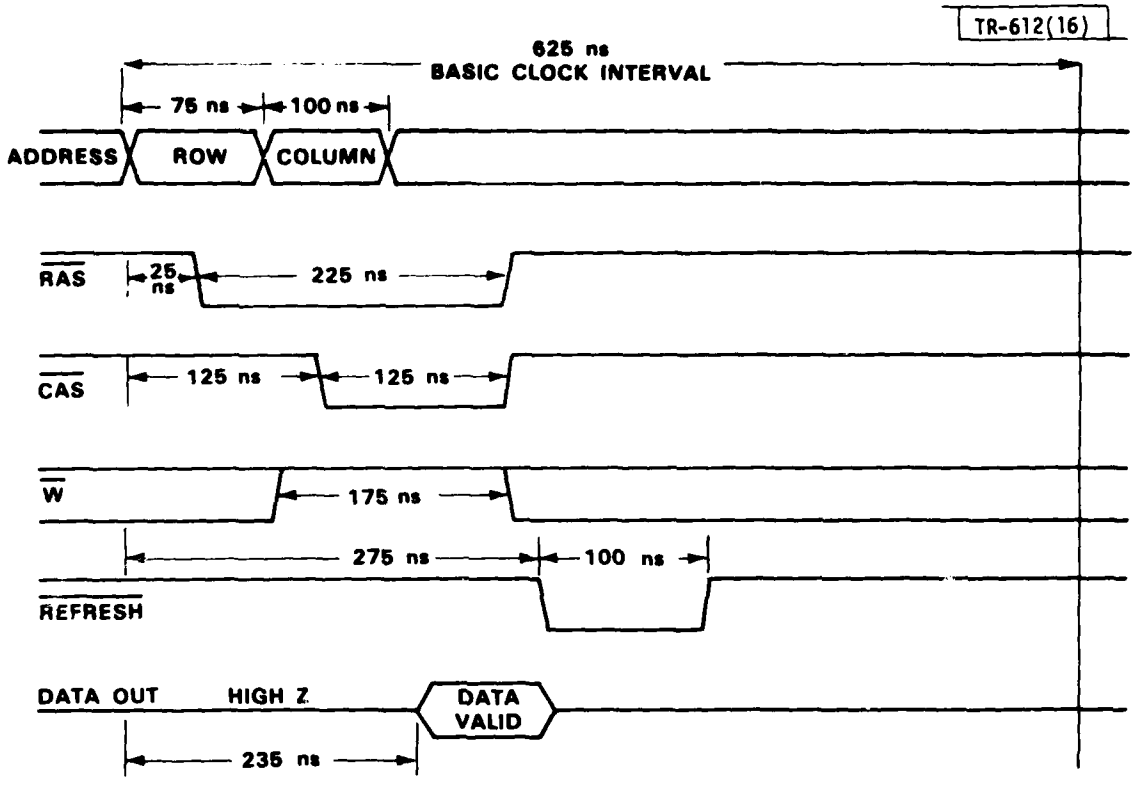


Fig. 16. Read operation timing.

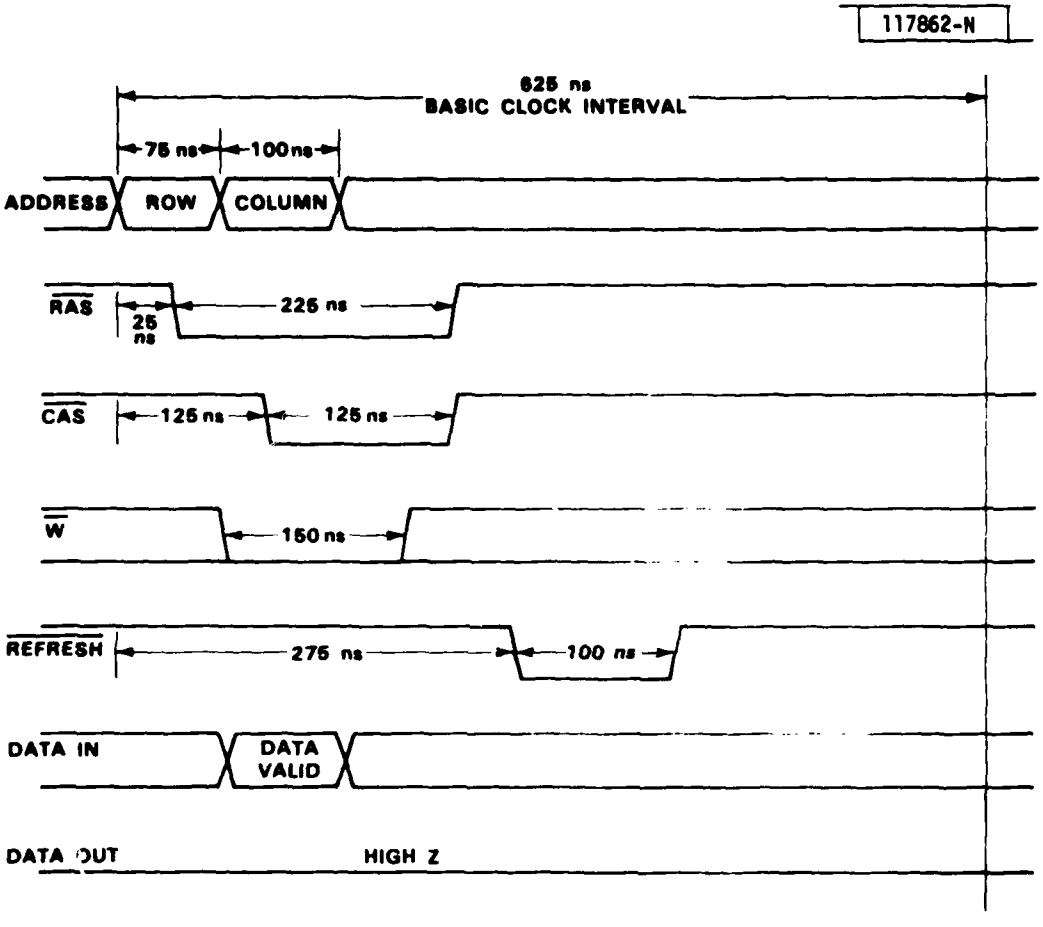


Fig. 17. Write operation timing.

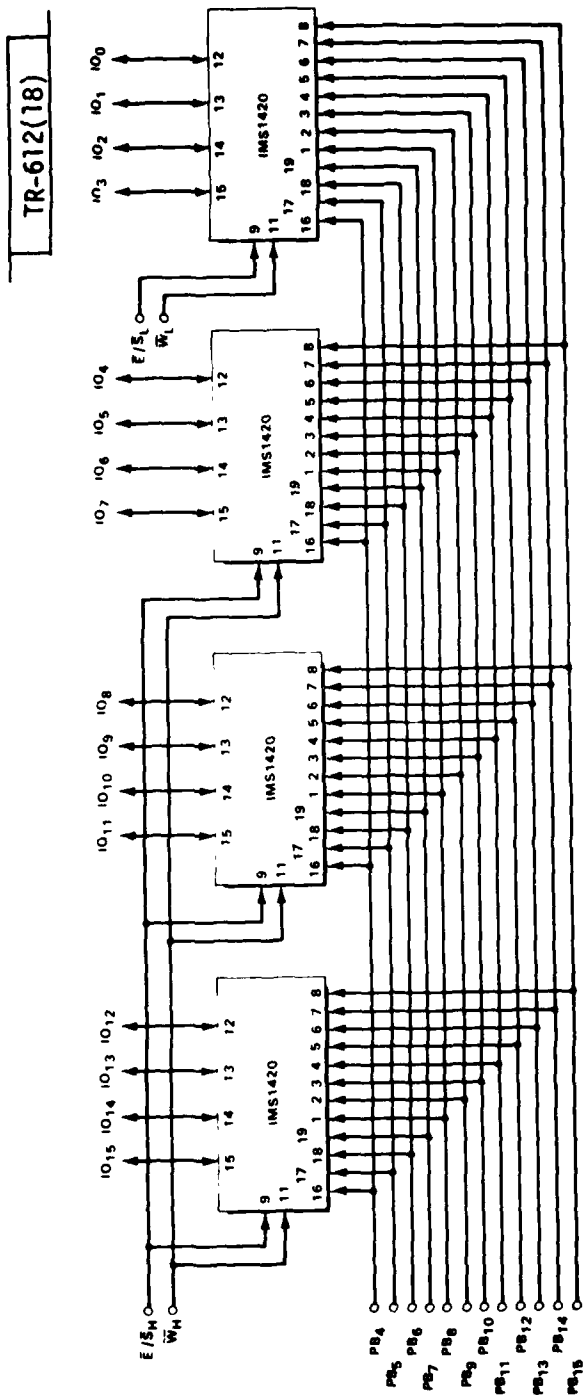


Fig. 18. Auxiliary memory complex.

buffer-full bit, and a spare bit. The parity bit is used in the Parity Detection Logic (Fig. 19) to provide a status bit to the PIO which is programmed to interrupt the CPU to allow the UMC Z80 to take corrective action in the event of a detected parity error.

The Auxiliary Pointer Memory Address Register (Fig. 20) is used to buffer the address bus contents in special situations where a different address component is required for the Pointer Memory than is currently appearing on the address bus for use by the Main Memory. The Auxiliary Memory Input/Output Register shown in Fig. 21 is used to buffer the memory inputs and outputs and to direct the data flow to and from the data bus of the MBM architecture.

The Auxiliary Pointer Memory is addressed every Main Memory cycle using the current address on the address bus to invoke an Auxiliary Pointer Memory read operation. The data from this read operation are used only when an end-of-block condition is detected, at which time the control signals from the Finite State Machine steer the Pointer Memory data to the Address Register Input Multiplexer. This is then used as the new address for the ongoing operations in Main Memory.

Timing for the Auxiliary Pointer Memory Static RAMs is shown in Fig. 22.

2.8. Address Register File/Counter Complex

The Address Register File/Counter complex is shown in Fig. 23. The principal elements of the structure are a set of 4 quad 2:1 Multiplexers (74157) , a set of 4- 4 x 4 Register File chips (74LS670), and a set of 4 Synchronous Address Counters (74LS697). The function of the entire complex is to provide 16-bit addresses to the address bus for use by either the Main Memory or the Auxiliary Pointer Memory. The Register File provides a set of separate registers which preserve the address status for each UMC Z80 port access to the shared memory between arbitrated accesses to other UMC Z80 ports. The Register File allows simultaneous read and write operations to occur in different addresses,

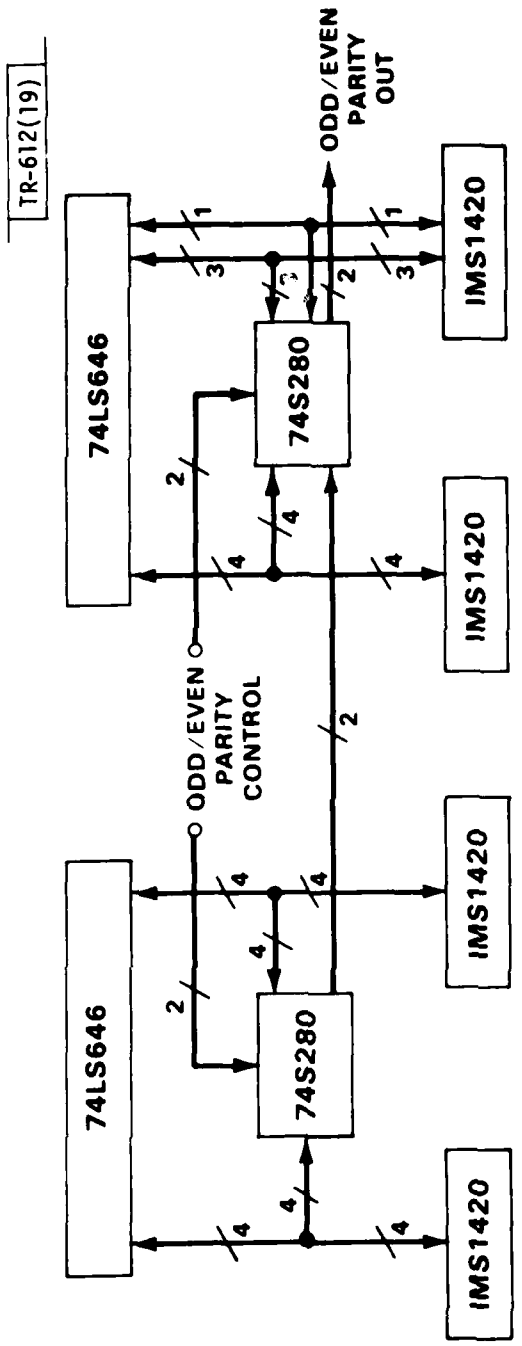


Fig. 19. Parity logic.

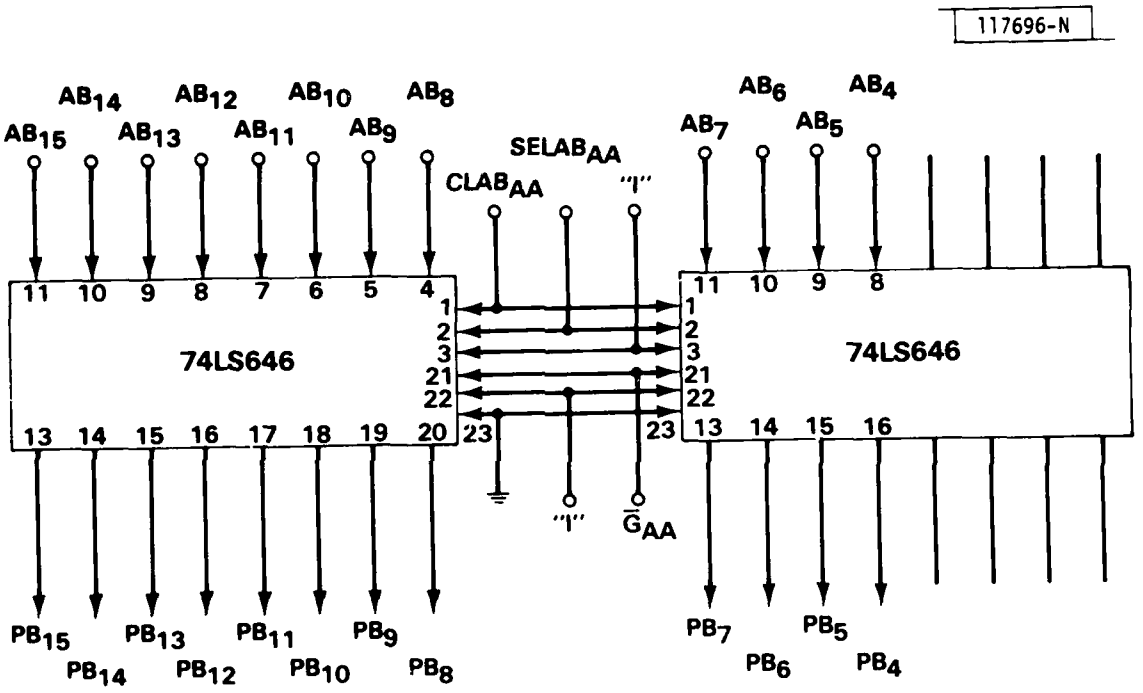


Fig. 20. Auxiliary memory address register.

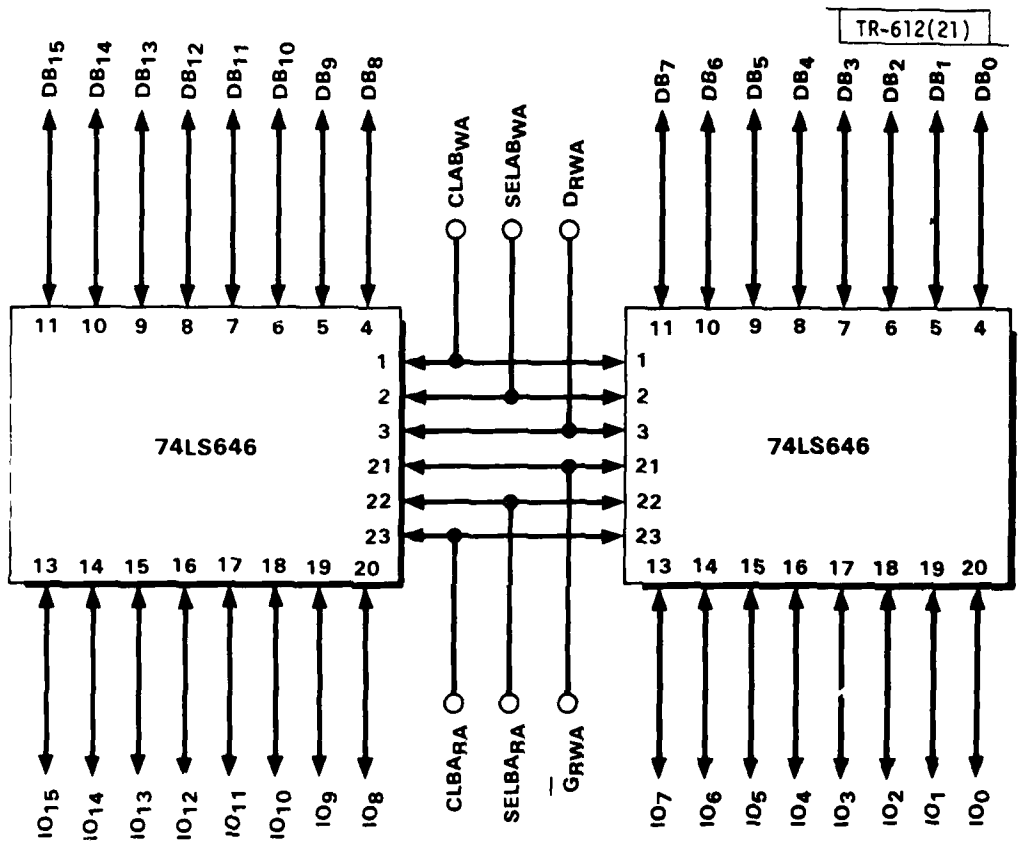


Fig. 21. Auxiliary memory I/O register.

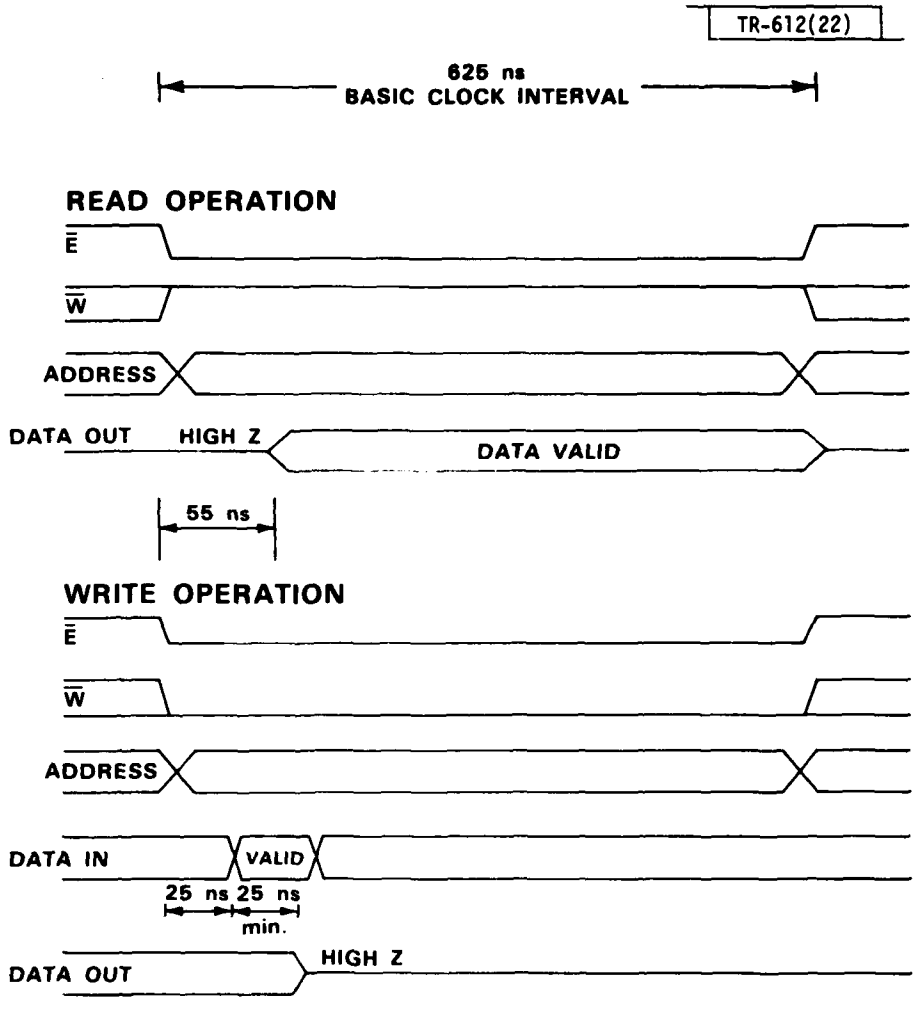


Fig. 22. Static RAM timing.

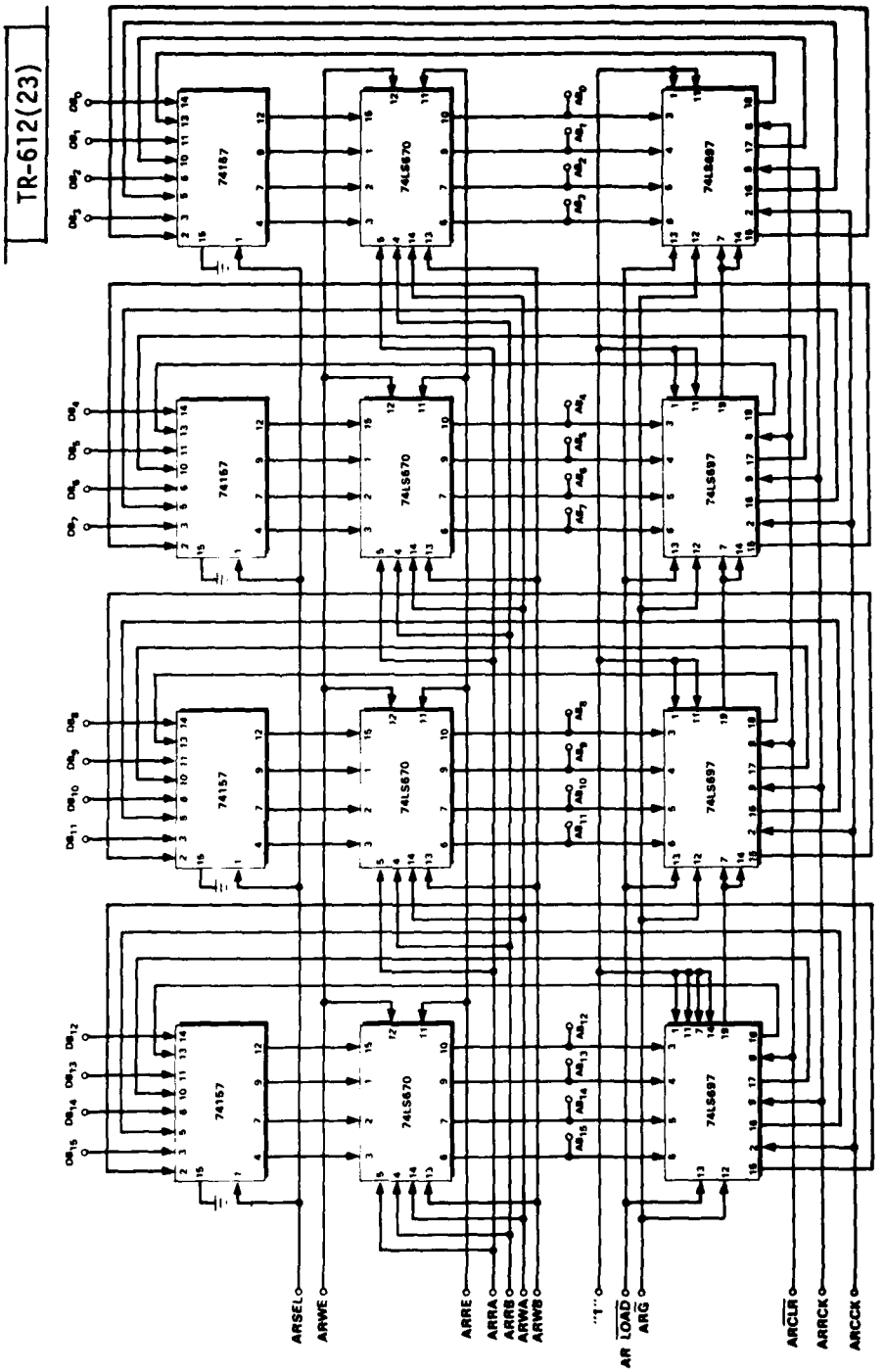


Fig. 23. Address register file/counter.

which facilitates the special ability to index the Address Register count within an ongoing memory access cycle and to apply the updated address to the memory address bus on the subsequent cycle. The 2:1 Multiplexer at the input to the Register File allows this function to occur without having to compete for the data bus within the same memory cycle. Inputs to the Address Register File are derived from either the counter output or from registers attached to the data bus. These include the Free List Register and the Input/Output register of the Auxiliary Pointer Memory which provides an address which is fetched from the Auxiliary Pointer Memory at the end of a 16-word data block in Main Memory. The end-of-block indication is derived from the Address Counter and is provided to the Finite State Machine as a **branching parameter.**

2.9. Length Register File/Counter Complex

The Length Register File/Counter complex (Fig. 24) is similar in design to the Address Register File/Counter with the exception that, whereas the Address Counter is incremented with each new memory access, the Length Counter is decremented in order to facilitate a packet word count detection when the counter reaches zero. This length=0 status is provided to the Finite State Machine as a branching parameter which signifies the end of a read or write operation when the number of words specified by the length parameter has been accessed in Main Memory.

2.10. Read Pointer Register/File

The Read Pointer Register/File (Fig. 25) is composed of a set of 4- 4 x 4 Register Files (74LS670) which provide a register storage mechanism for the read pointer which serves to indicate the starting location of packets stored in the Main Memory. This register is used only in conjunction with a read/release operation which then requires the release of the read storage locations to the free list. This operation is effected by transferring the Read Pointer Register contents to the free list after a read/release operation has been completed.

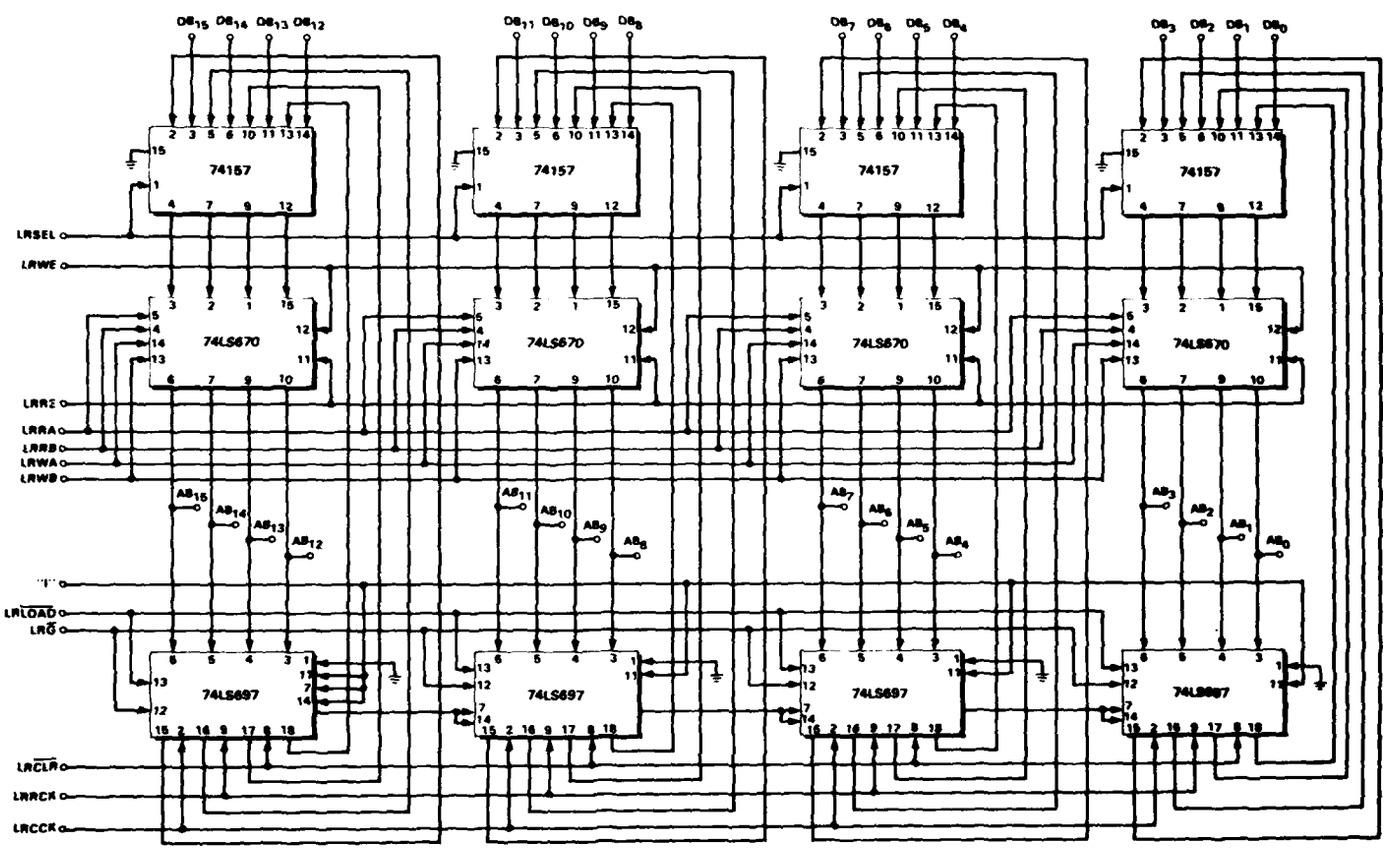


Fig. 24. Length register file/counter.

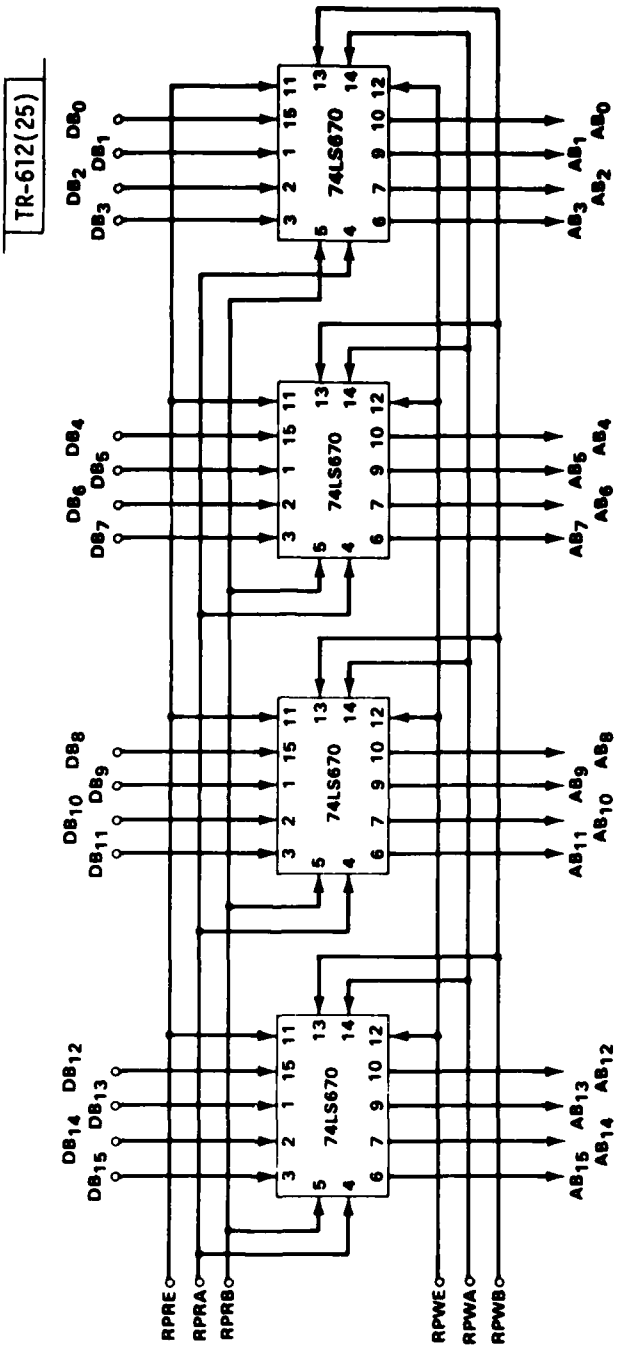


Fig. 25. Read pointer register file.

2.11. Free List Register

The Free List Register (Fig. 26) is a single register which is used to store the current location of the *free list* which represents the top of a stringed list of memory block locations that are available for write operations. The register is implemented using a pair of 74LS646 octal bus transceivers and registers which allow two-way communication between the data bus and address bus of the MBM.

2.12. Basic System Timing

The block diagram of the basic system timing is shown in Fig. 27. The basic system clock interval is 625 ns and a synchronous counter driven by a 25 ns master oscillator subdivides each 625 ns Main Memory cycle into intervals which are decoded to provide appropriate timing signals for the various register transfers that must occur within the architecture. Special control signals for the Main and Auxiliary Memory complexes are provided by memory timing logic which generates the timing sequences shown in Figs. 16, 17, and 22.

The system clock pulses are gated by controls emanating from the Finite State Machine control outputs and from the Bus Arbitration Logic, where appropriate, to effect the arbitration of grants to the various UMC Z80s competing for simultaneous access to the shared memory.

3. MEMORY MANAGEMENT LOGIC

3.1. Basic Philosophy

The MBM employs an intelligent controller stratagem which provides all of the internal bookkeeping necessary to create a storage and retrieval scenario for the shared Main Memory which relieves each of the 4 UMC Z80s requesting access to the memory of the overhead associated with block transfer management.

The basic strategy is to subdivide the Main Memory array of 64K storage locations into blocks of 16-word *strings* in which packets of data emanating from

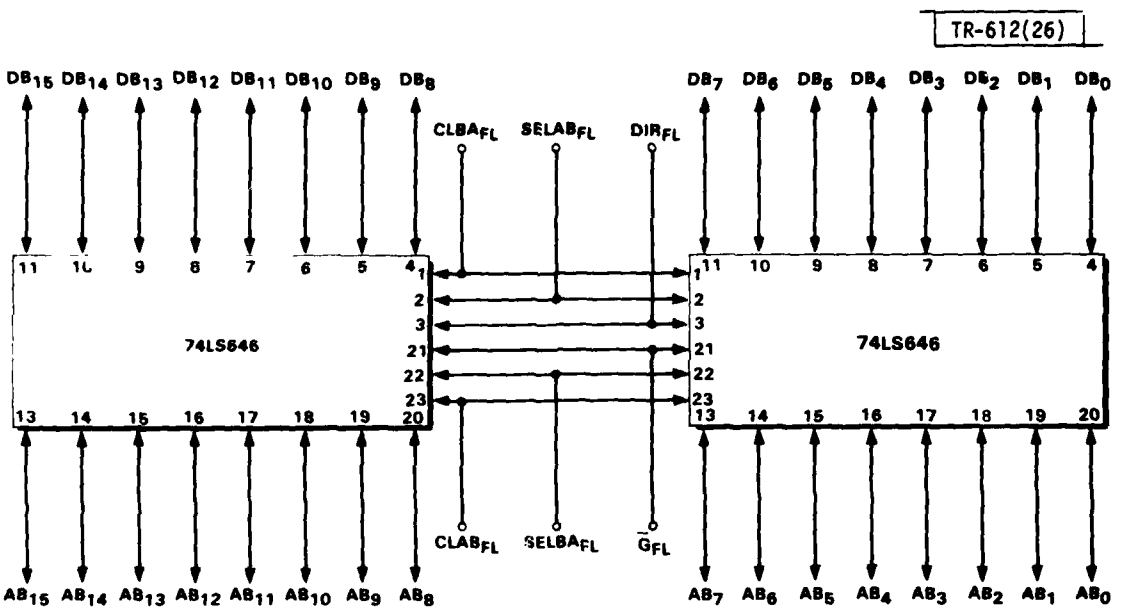


Fig. 26. Free list register.

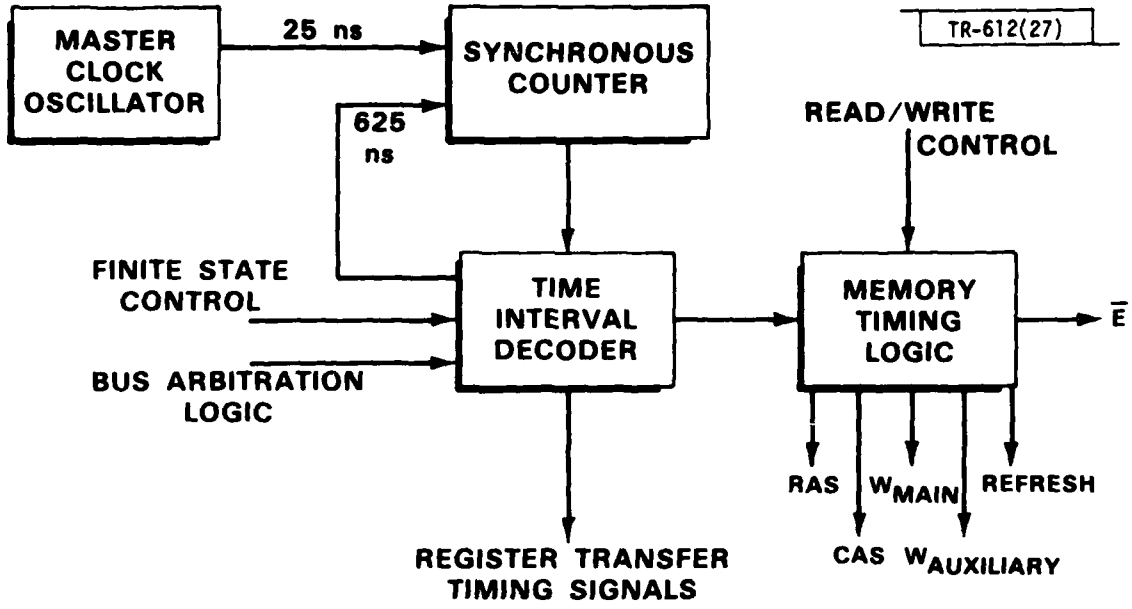


Fig. 27. System timing logic.

each UMC Z80 complex are stored and retrieved. Since the Main Memory complex is to be shared among 4 of the UMC Z80s there is a continual exchange of available locations as data are written, and subsequently read, in the memory. There are a total of 4K blocks in the 64K Main Memory array.

Since read and write operations may be totally interleaved due to bus arbitration among the 4 UMC Z80s it is necessary to provide some mechanism for keeping track of the free blocks in Main Memory as new locations are used in the writing process and formerly used locations are made available in the read/release process. The vehicle used for this bookkeeping is the Free List Register. During the initialization procedure the Free List Register contents are set to 0 and, thus, the pointer to the first free location in Main Memory is set at the conceptual top of the Main Memory array. The initialized contents of the Auxiliary Pointer Memory (Fig. 5) consist of a stored list of 12-bit addresses which point to stringed 16-word blocks in Main Memory.

As read and write operations are interspersed in the Main Memory due to bus grants to the various UMC Z80 complexes the Finite State Machine which controls these operations contains a branch point when the Address Counter reaches the end of a 16-word block. At this branch point the new address input supplied to the Address Register File is derived from the Read Pointer Memory contents whose address consists of the current block address on the address bus. In the case of an initially unwritten Main Memory array (startup condition) the next block would be the contiguous address block; however, in the general case of interspersed write and read operations the next block location will be different. In order to understand this situation Figs. 28(a) and 28(b) present a scenario which traces a sequence of write and read operations through the Main and Auxiliary Memory complexes. Fig. 28(a) depicts a situation whereby three separate packets of data have been stored in Main Memory from three different UMC Z80 complexes. These packets are labeled A, B, and C. To simplify the dis-

TR-612(28a)

16 WORD BLOCK ADDRESS	MAIN MEMORY DATA
0	BLOCK A
1	BLOCK A
2	BLOCK A
3	BLOCK A
4	BLOCK B
5	BLOCK B
6	BLOCK B
7	BLOCK C
8	BLOCK C
9	BLOCK C
4095	

ADDRESS	AUXILIARY POINTER DATA	SPARE	BUFFER FULL	STRING TERMINATOR	PARITY
0	1	0	0	0	1
1	2	0	0	0	1
2	3	0	0	0	0
3	4	0	0	1	0
4	5	0	0	0	0
5	6	0	0	0	0
6	7	0	0	1	0
7	8	0	0	0	1
8	9	0	0	0	0
9	10	0	0	1	1
4095	0	0	1	0	1

0

READ POINTER REGISTER

10

FREE LIST REGISTER

Fig. 28(a). Memory management scenario.

TR-612(28b)

16 WORD BLOCK ADDRESS	MAIN MEMORY DATA
0	BLOCK A
1	BLOCK A
2	BLOCK A
3	BLOCK A
4	BLOCK D
5	BLOCK D
6	BLOCK D
7	BLOCK C
8	BLOCK C
9	BLOCK C
10	BLOCK D
11	BLOCK D
12	BLOCK D
≈	≈
4095	

4

READ POINTER REGISTER

ADDRESS	AUXILIARY POINTER DATA	SPARE	BUFFER FULL	STRING TERMINATOR	PARITY
0	1	0	0	0	1
1	2	0	0	0	1
2	3	0	0	0	0
3	4	0	0	1	0
4	5	0	0	0	0
5	6	0	0	0	0
6	10	0	0	0	0
7	8	0	0	0	1
8	9	0	0	0	0
9	10	0	0	1	1
10	11	0	0	0	1
11	12	0	0	0	0
12	13	0	0	1	0
≈	≈	≈	≈	≈	≈
4095	0	0	1	0	1

13

FREE LIST REGISTER

Fig. 28(b). Memory management scenario.

cussion it is assumed that these three packets were written at different times rather than having been written under a more general situation of interleaved access under control of the Bus Arbitration Logic. Packets A, B, and C are shown to occupy 4, 3, and 3 blocks of 16 words, respectively. In this case the contents of the Auxiliary Pointer Memory are shown to be undisturbed over the initial conditions for this memory as shown in Fig. 5, except for the fact that each block of packets has attached a string terminator in the final block location to flag the end of that packet. Note that the Free List Register at this point contains the address pointer 10 which points to the next free location available for writing in the Main Memory complex. The Read Pointer Register value at this point is equal to 0, which represents the initial contents of that register since no blocks have been read at this point for purposes of the discussion. Fig. 28(b) depicts a rearranged configuration of both the Main and Auxiliary Pointer Memory complexes which reflects a scenario whereby packet B (which is composed of 3 blocks of 16 words) has been read and released from Main Memory and replaced by a new packet string (packet D) comprised of 6 blocks of 16 words. The revised arrangement of these 6 blocks shows them interspersed in the three vacated locations between packets A and C and continued at the end of the packet C location. The Read Pointer Register contains 4 at this point, which reflects the fact that the last packet read (B) began at block location 4 in Main Memory. The Auxiliary Pointer Memory contents have been altered by the Memory Management Logic to show the removed string terminator at location 6 and the added string terminator at location 12. In addition to the string terminator revision the pointer in location 6 has been changed from 7 (which would have been the address of the next consecutive block in Main Memory) to 10, which was the previous free list value containing the next available writable location in Main Memory. The new Free List Register value has been modified to contain 13, which is the first location of unoccupied Main Memory after the rear-

ranged packet strings.

The Memory Management Logic is effected under the control of a Finite State Machine which steps the architecture through the read and write operations in the MBM by supplying control states for each step that determine which of the registers should be connected to the address and data busses of the architecture. The following discussion describes the functional operations which comprise the read and write operations. The discussion assumes a request for access from a particular UMC Z80 complex, and all transfers are made to and from that complex. The influence of bus arbitration is not detailed in this context other than to signify that various operations are subject to port assignments emanating from the Bus Arbitration Logic.

4. READ OPERATION

The read operation in the MBM may be either a read/restore or a read/release. The read/restore operation is generally used in a conferencing scenario where it is necessary to send more than one copy of a packet out to other networks. The read-release operation is the normal type of read. Fig. 29 shows the flow chart of the memory read operation. The read operation consists of the following transfers:

4.1. Transfer 2 bytes from the UMC Z80 data bus to the Input Bus Latches. These bytes represent a 16-bit length parameter which specifies the number of words in the packet to be read.

4.2. Signal the PIO that a read operation is being requested in the shared memory. This request is passed on to the Bus Arbitration Logic as a request for access. The PIO signifies the type of read operation to be performed (read/restore or read/release). The PIO also signals the interface that the length parameter is ready for transfer at this point.

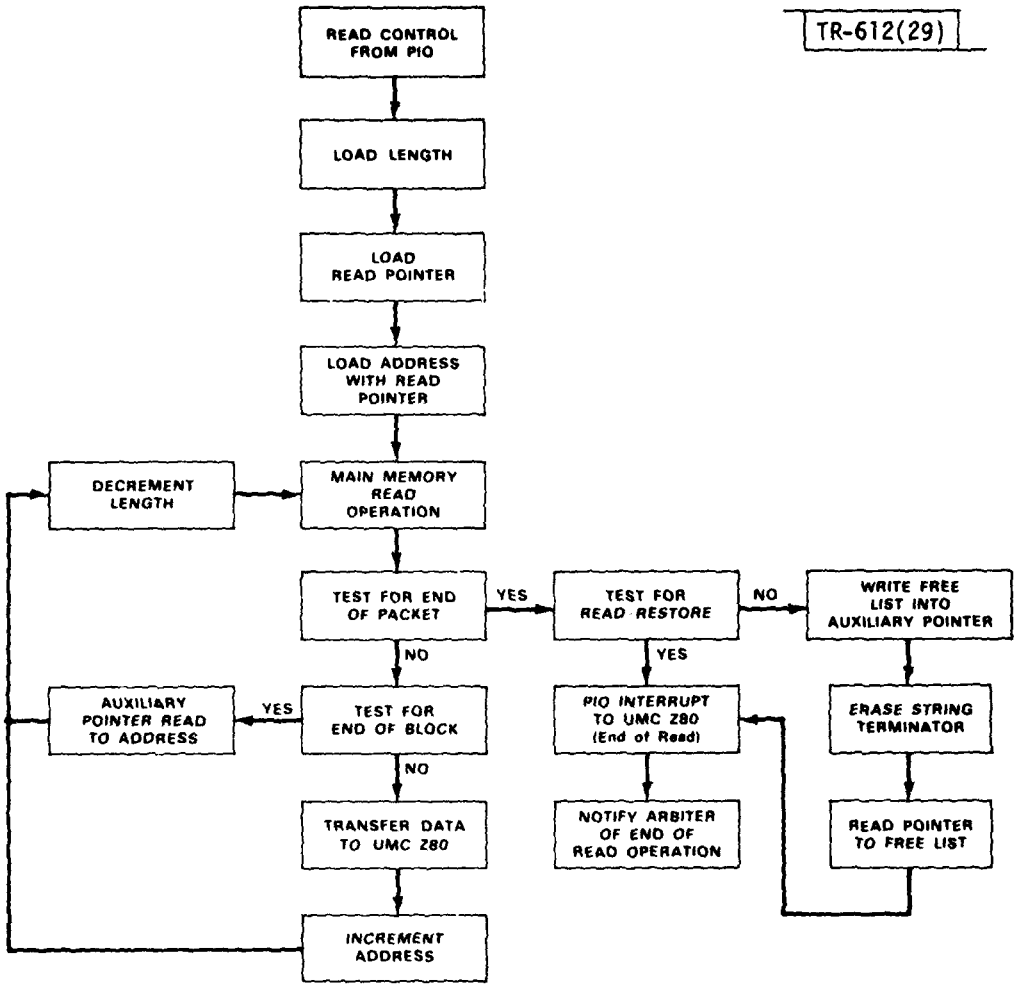


Fig. 29. Read flow chart.

4.3. When the pending read request is granted by the Bus Arbitration Logic, transfer the length parameter to the appropriate Length Register File. The appropriate file address is provided by the Bus Arbitration Logic which issues the grant signal to a particular port.

4.4. Transfer 2 bytes from the UMC Z80 data bus to the Input Latches. These bytes represent the 16-bit pointer address to the first location in Main Memory in which the packet to be read is stored.

4.5. Signal the PIO that the 16-bit read pointer is ready for transfer.

4.6. When granted access by the Bus Arbitration Logic, transfer the read pointer to the appropriate Read Pointer Register as well as to the appropriate Address Register File. The file address is again provided by the Bus Arbitration Logic.

4.7. Set up the DMA attached to the external UMC Z80 bus for the read operation and indicate its readiness to proceed when granted access by the Bus Arbitration Logic.

4.8. Proceed with the read operation under the control of the DMA handshake and the Bus Arbitration Logic. The handshake logic provides for the transfer of two UMC Z80 bytes for every MBM word. The read operation is characterized by the following functional transfers which are effected within a single 625 ns Main Memory cycle as governed by the read timing control:

4.8.1. Transfer the selected Address Register File contents to the address bus and to the Address Counter.

4.8.2. Transfer the selected Length Register File contents to the Length Counter.

4.8.3. Transfer the output of the Main Memory to the Output Latches of the selected UMC Z80 port.

4.8.4. Increment the Address Register Counter and write the updated address into the Address Register File at the current location.

4.8.5. Decrement the Length Register Counter and write the updated length into the Length Register File at the current location.

4.9. When the block counter reaches 15, the current Main Memory address is transferred to the Auxiliary Memory Address Register and a read operation is performed in the Auxiliary Memory at that address. The results of this read operation are transferred to the Address Register File at the current address of the arbitrated access. This register now contains the new block address as pointed to by the Auxiliary Pointer Memory.

4.10. When the Length Counter reaches 0, the Bus Arbitration Logic is notified and an interrupt bit to the appropriate PIO is set to signal the termination of the read operation.

4.11. If the operation is a read/restore no further action is necessary after termination of the read and notification to the Arbiter Logic. If the operation is a read/release then the current address of the read operation is used to address the Auxiliary Pointer Memory for a write operation. The inputs for the write operation are derived from the contents of the Free List Register for the 12 MSB and from a hard-wired input for the 4 LSB which erases the string terminator.

4.12. Upon completion of the write operation to the Auxiliary Pointer Memory in the case of a read/release operation the Read Pointer Register File contents are transferred to the Free List Register. This completes the read/release operation. These read operations are performed subject to control by the Bus Arbitration Logic.

5. WRITE OPERATION

The write operation flow chart is shown in Fig. 30. The write operation consists of the following transfers:

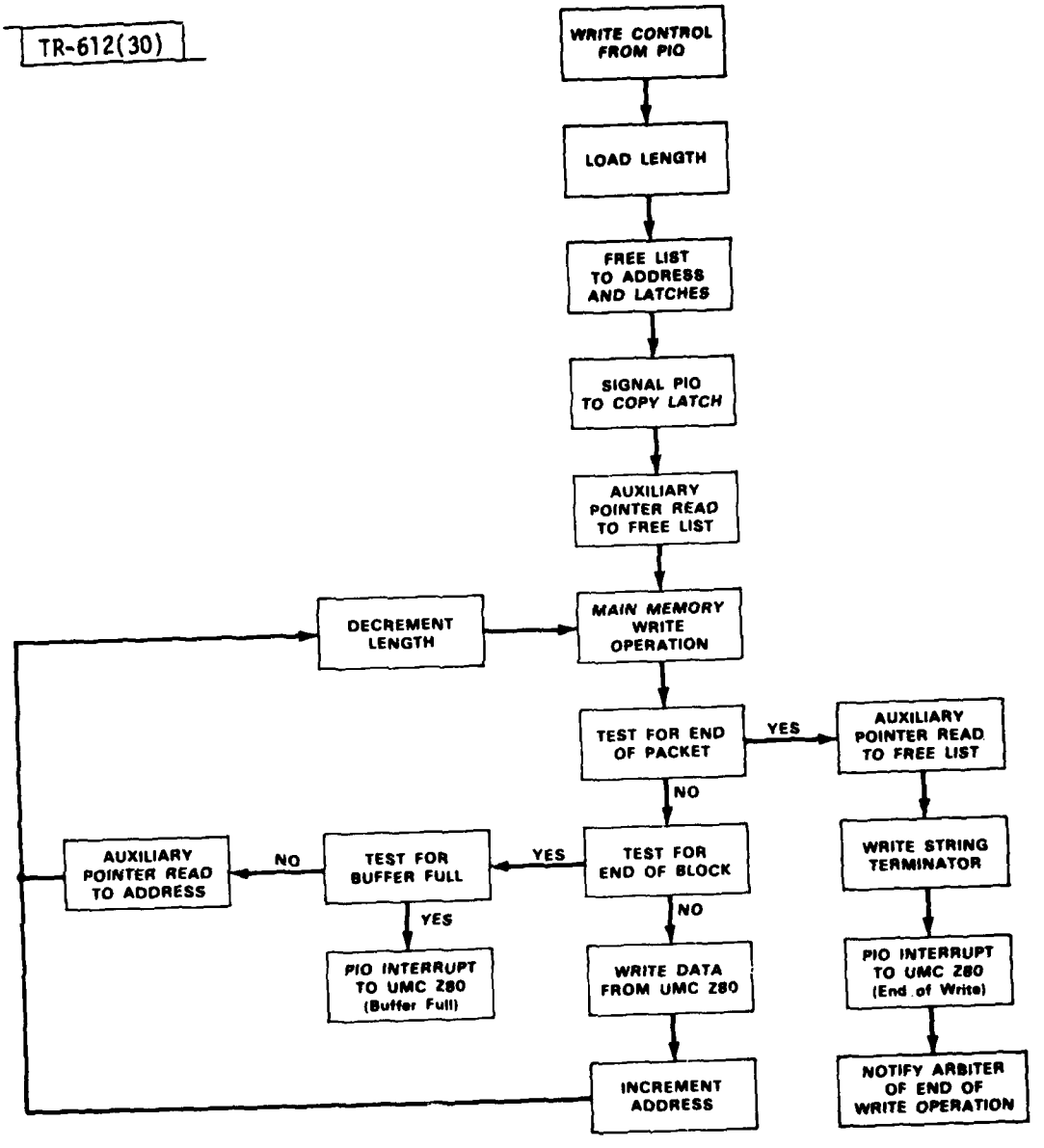


Fig. 30. Write flow chart.

5.1. Transfer two bytes from the UMC Z80 data bus to the Input Data Latches from that bus. These bytes represent a 16-bit length parameter which specifies the number of words in the packet to be written.

5.2. Signal the PIO that a write operation is being requested in the shared memory. This request is passed on to the Bus Arbitration Logic as a request for access. The PIO also signifies that the length parameter is ready for transfer at this point.

5.3. When the pending write request is granted by the Bus Arbitration Logic, transfer the length parameter to the appropriate Length Register File. The appropriate file address is provided by the Bus Arbitration Logic which issues the grant signal to a particular port.

5.4. When the bus is again granted to this particular port, transfer the Free List Register to the Output Data Latches as well as to the Address Register File for this port address. This cycle also provides a signal which enables a PIO interrupt to the UMC Z80 to indicate that the Free List Register has been loaded into the Output Latches to be copied as soon as it is programmed by the UMC Z80 to do so. This value then becomes attached to the packet header as the pointer to the first address which has been assigned in Main Memory to that packet. During this same cycle the address which has just been transferred from the Free List Register to the Address Register File is placed on the address bus and used to address the Auxiliary Pointer Memory for a read operation. The read output of the Auxiliary Pointer Memory is transferred to the Free List Register such that it now contains the pointer to the next free block in Main Memory in preparation for any subsequent write operation to be granted by the Bus Arbitration Logic.

5.5. Set up the DMA attached to the external UMC Z80 bus for the write operation and indicate its readiness to proceed when granted access by the Bus Arbitration Logic.

5.6. Proceed with the write operation under the control of the DMA handshake and the Bus Arbitration Logic. The handshake logic provides for the transfer of two UMC Z80 bytes for every MBM word. The write operation is characterized by the following functional transfers which are effected within a single 625 ns Main Memory cycle as governed by the write timing control:

5.6.1. Transfer the selected Address Register File contents to the address bus and to the Address Counter.

5.6.2. Transfer the selected Length Register File contents to the Length Counter.

5.6.3. Transfer the Input Data Latch contents to the data bus to provide the input to the Main Memory.

5.6.4. Increment the Address Register Counter and write the updated address into the Address Register File at the current location.

5.6.5. Decrement the Length Register Counter and write the updated length into the Length Register File at the current location.

5.7. When the Block Counter reaches 15, the current Free List Register is transferred to the Auxiliary Pointer Memory Address Register which then provides an address for the Auxiliary Pointer Memory. A read operation is performed at this address in the Pointer Memory. The read output of the Pointer Memory is transferred to the Address Register File and to the Free List Register. The write operation then proceeds under the control of the DMA handshaking and the Bus Arbitration Logic.

5.8. When the Length Counter reaches 0, the current Address Register File is transferred to the Auxiliary Pointer Address Register and a write operation is performed in the Pointer Memory which attaches a string terminator in this memory location to signify the end of the written packet for identification during subsequent read operations.

5.9. When the Pointer Memory write operation is completed the Arbiter is notified and the an interrupt is engendered in the PIO which signals the UMC Z80 that the write operation has been completed.

6. FINITE STATE MACHINE

The Finite State Machine shown in Fig. 3 provides the control signals which govern the transfer of data between the various registers and the main and auxiliary memory modules of the MBM architecture. The controller is implemented with a set of PROMs which output a set of control signals and next state parameters necessary to effect the memory read and write flow charts shown in Figs. 29 and 30.

The address inputs to the PROMs are made up of two basic components. One component of the address is provided by next state controls emanating from one of the PROMs. These controls are stored in a set of 4 x 4 register files in order to preserve the states of the address for the Finite State Machine between arbitrated accesses to the various UMC Z80 ports. The address of the PROMs for any given MBM cycle is dictated by which of the 4 register files is selected by the 2-bit port address issued from the Bus Arbitration Logic.

The second component of the PROM addresses is provided by the current states of the critical branching parameters which are composed of:

1. The length = 0 signal
2. The block counter = 15 signal
3. The buffer-full flag

The branching parameters are provided as a direct input to the address register file of the Finite State Controller, whereas the the next state control outputs of the PROM are delayed in a set of latches by one full main memory cycle such that an effective pipelining is achieved whereby the branching takes place in a prioritized fashion as soon as the branching parameters are detected.

The architecture control signals are used to gate timing pulses generated by the system timing logic which govern the relative times at which register transfers occur.

7. BUS ARBITRATION LOGIC

The Bus Arbitration Logic shown in Fig. 31 is implemented as a PROM-based controller whose address inputs are derived from a combination of current requests delivered through the PIOs of the 4 UMC Z80 ports attached to the MBM and the addresses of the past three bus grants. The previous bus addresses are stored in shift registers which provide a record of past bus grants for application of the selected arbitration algorithm stored in the PROMs. Since the priority algorithm for bus arbitration is implemented in PROM it may be easily modified as the system design evolves. The simplest arbitration algorithm consists of a polling arrangement whereby a conceptual 4 cycle memory span of 2.5 us allocates accesses in some arbitrarily chosen order such that each port is granted access to the main memory at least once every 2.5 us. The sequence is preserved in that order to assure that no one port receives successive bus grants any more frequently than 2.5 us for the case of all four ports requesting simultaneous access. Since asynchronous requests from the various ports are synchronized to the MBM system clocks they may appear to arrive simultaneously and the bus arbitration algorithm sorts the priority in the larger context of polled access. For the case of a single port requesting access the arbitration is overlaid by the DMA handshaking logic such that it is granted access as fast as the DMA can regulate the data transfers.

8. ENGINEERING DETAILS

The MBM is to be constructed using two PROTOHEX Wirewrap Boards. This board employs a special geometry to allow placement of assorted package sizes with 0.3, 0.4, or 0.6 in. lead spacing. The capacity is 144-14 pin packages, but the

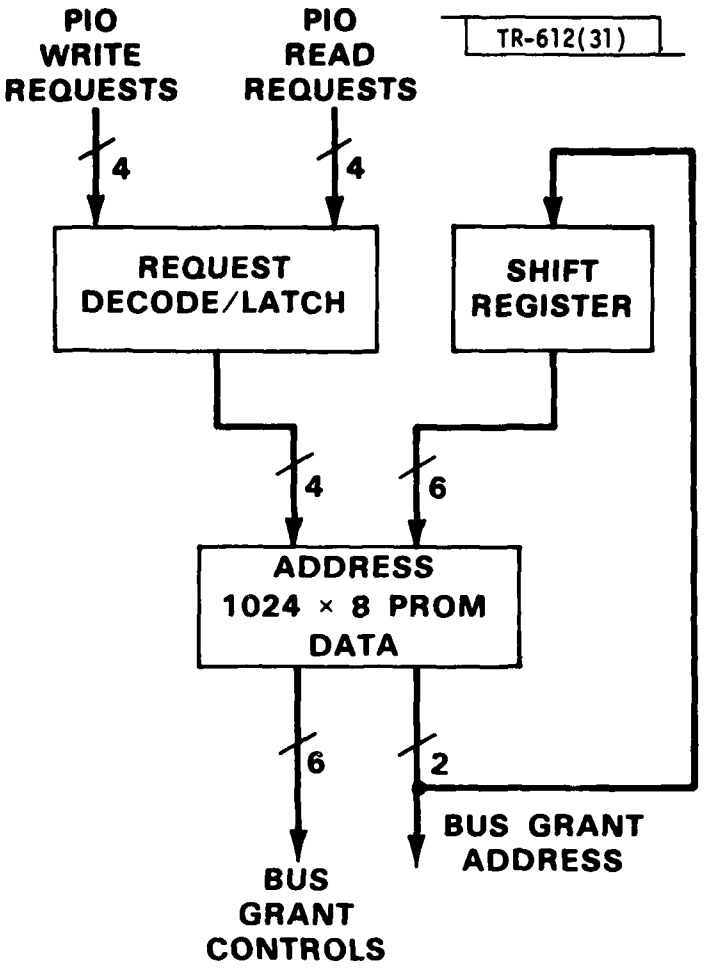


Fig. 31. Bus arbitration logic.

DIP count is modified accordingly as other pin counts are used in the universal geometry.

The PROTOHEX board is designed to plug into the UNIBUS on one side and has a set of four 50-pin edge connectors on the other side which makes it suitable for connection to other UMC Z80 boards that employ the same connector. Power is supplied through the UNIBUS backplane and is distributed on a special layer of the PROTOHEX board. High and low frequency decoupling capacitors are distributed throughout the board at all DIP locations to facilitate noise rejection. The board is designed so that wirewrap pins protrude through the component side and, therefore, only a single hex-width slot is required to house each board in the UNIBUS backplane. This is a very critical requirement in view of the limited board complement which can be accommodated in a particular Miniconcentrator configuration. Because of this limited backplane space a special board arrangement has been designed to make use of backplane slots which are presently unused due to a jumper board which is inserted in several of the slots to join subrack sections together. Since these jumper boards occupy only two of the six hex slots in the backplane a special sectored board can be used which is cut to allow insertion of jumper boards along with the PROTOHEX board in the same backplane slot.

The partitioning of the full MBM is guided by the dual requirements for full UMC Z80 connector utilization and minimal interboard cabling. In view of these requirements the basic board partitioning arrangement is shown in Fig. 32. The two PROTOHEX boards are partitioned so that two UMC Z80s can be connected to the four 50-pin connectors on each board. This arrangement accommodates all of the interboard cabling between UMC Z80s and the MBM since each connection to a UMC Z80 requires two 50-pin flat conductor cables. This divides the architecture of the MBM vertically across the UMC Z80 interfaces, and the detailed division of the subsections of the overall architecture is depicted in Fig. 32.

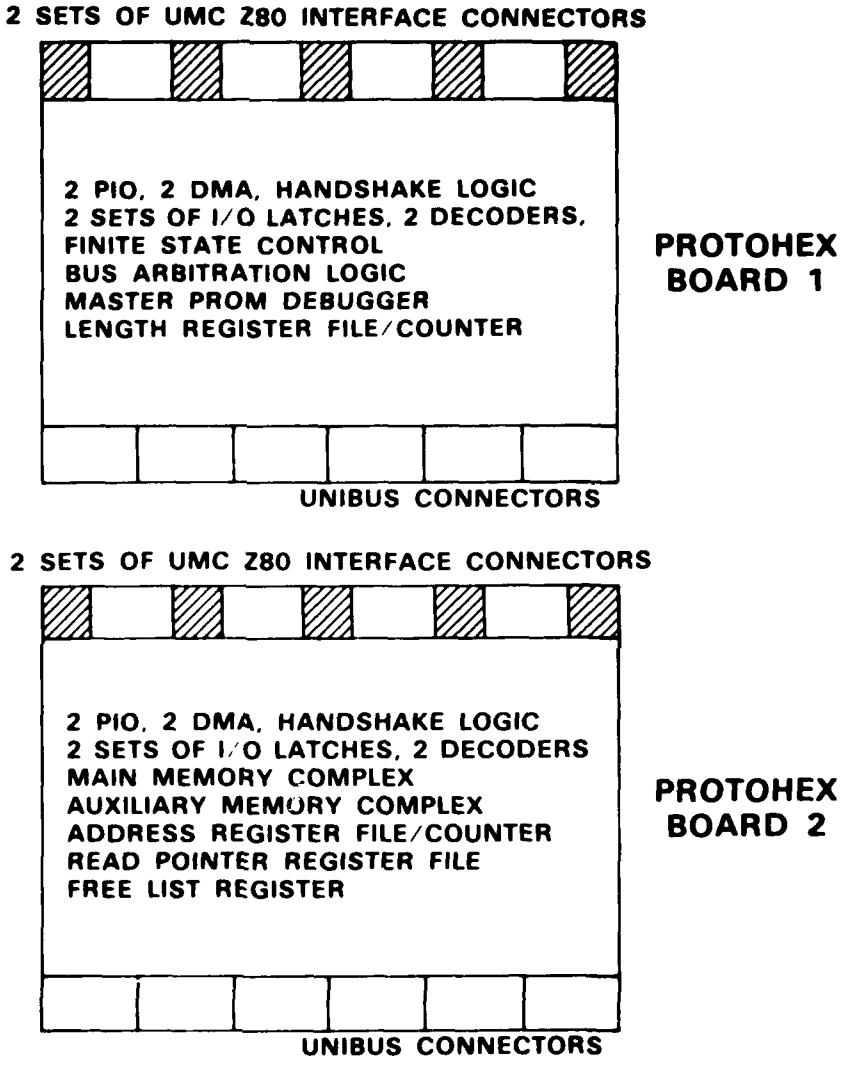


Fig. 32. Board partitioning.

The interboard cabling between the two PROTOHEX boards is provided through 16-pin DIP connectors mounted on each board. These connectors contain a set of twisted pair cables which are attached to line drivers and receivers for balanced twisted pair communication between the two boards. Fig. 33 shows the signals which are transmitted over the interboard cables.

The MBM has been designed to fulfill a current need for improved throughput in the total Miniconcentrator environment. The use of a bussed structure and programmed control of the architecture provides a flexible means of updating the design to accommodate new system requirements. This design is therefore compatible with an expanding experimental complex which is embodied in the WB SATNET.

9. ACKNOWLEDGEMENT

The author wishes to acknowledge the many helpful discussions and suggestions offered by J.W.Forgie throughout the course of this design.

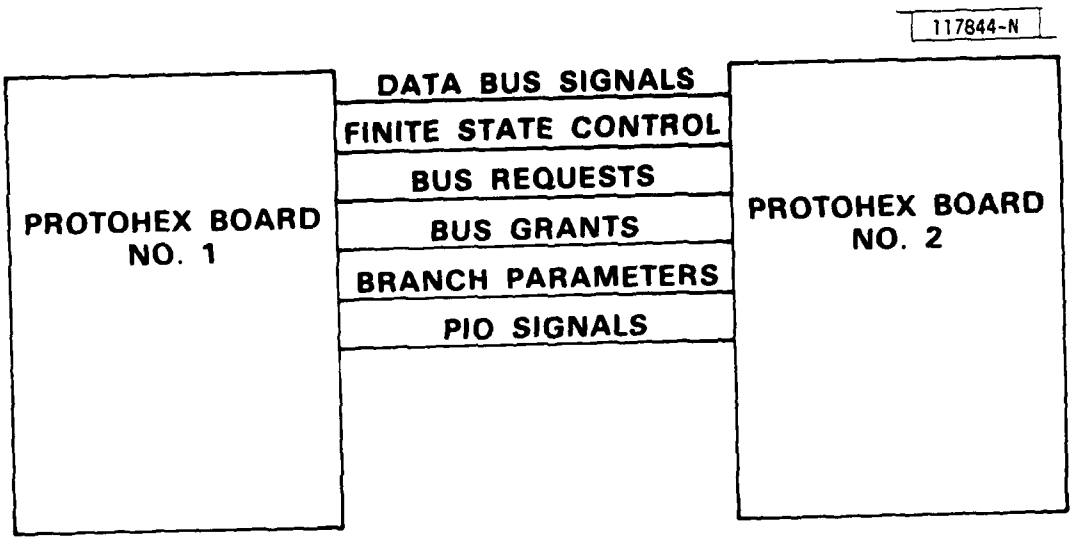


Fig. 33. Interboard cabling.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-82-056	2. GOVT ACCESSION NO. AD-A119 157	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A Multiport Buffer Memory for an Internet Packet Voice/Data Gateway		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER Technical Report 612
7. AUTHOR(s) Vincent J. Sferrino		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element Nos. 61101E and 62708E Project Nos. 2D10 and 2T10 DARPA Order 3673
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 14 July 1982 13. NUMBER OF PAGES 68
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB, MA 01731		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

multiport buffer memory
microprocessors
wideband satellite network
register file
bus arbitration
finite state machine
memory management

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes the design of a Multiport Buffer Memory to be used in an internet packet voice/data gateway for an experimental wideband satellite network. The gateway consists of a central processor and a number of microprocessor-controlled interfaces to local networks at each site and to the satellite network. The Multiport Buffer Memory is accessible from any of these microprocessors, and increases system throughput by allowing the bulk of the packet traffic between the networks to pass through it without having to pass through the central processor.

The shared memory consists of a main memory organized as 64K words x 16 bits and an auxiliary pointer memory of 4K words x 16 bits. The Multiport Buffer Memory complex includes an automated intelligent memory management strategy which makes the memory resource allocation transparent to the microprocessor-based systems which share the memory.

DD FORM 1 Jan 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*